UNIVERSIDADE DE LISBOA

FACULDADE DE CIÊNCIAS

DEPARTAMENTO DE INFORMÁTICA

PERSONAGENS VIRTUAIS AUTÓNOMOS EM CENÁRIOS TRIDIMENSIONAIS

Willian Xavier da Silva

MESTRADO EM ENGENHARIA INFORMÁTICA
ESPECIALIZAÇÃO EM INTERAÇÃO E CONHECIMENTO

Trabalho de projeto orientado por:
Prof.ª Doutora Ana Paula Boler Cláudio
e Prof.ª Doutora Maria Beatriz Duarte Pereira do Carmo.

2019

Agradecimentos

Um trabalho de mestrado é uma longa viagem, que inclui uma trajetória permeada por inúmeros desafios, tristezas, incertezas, alegrias e muitos percalços pelo caminho, por este motivo, o contributo de várias pessoas se tornam indispensáveis para encontrar o melhor caminho em cada momento desta trajetória.

Trilhar este caminho só foi possível com apoio e a força de várias pessoas, a quem dedico especialmente este projeto de vida.

Um agradecimento muito especial à minha família, em especial à minha mãe e ao meu pai pelo incentivo na busca de conhecimento, por me acompanhar e por terem proporcionado as condições necessárias à realização de todo o meu percurso académico. Sem eles, não teria sido possível.

Agradeço às minhas orientadoras, a Professora Doutora Ana Paula Boler Cláudio e a Professora Doutora Maria Beatriz Duarte Pereira do Carmo, pela orientação exemplar pautada por um elevado e rigoroso nível científico, um empenho inexcedível durante todo o percurso deste trabalho, onde tiveram um papel fundamental em todas as etapas subjacentes para a realização do trabalho. Muito obrigado!

Agradeço ao Professor Doutor Rui Filipe Antunes pelo seu contributo e disponibilidade ao longo deste trabalho.

Agradeço aos Professores e Professoras do Departamento de Informática da Faculdade de Ciências da Universidade de Lisboa durante o percurso académico, pelos ensinamentos transmitidos com eficiência e dedicação.

Não poderia deixar de agradecer a minha eterna companheira Cristina Dominguez por ter caminhado ao meu lado, pela sua paciência, compreensão e motivação incondicional. Muito Obrigado!

Agradeço aos meus colegas que fizeram parte deste meu percurso de formação académica e também pela convivência.

A todos o meu sincero e profundo muito obrigado!

À minha família, amigos e namorada.

Resumo

Nas últimas décadas, a tecnologia tem permitido criar mundos virtuais, em particular recriações de civilizações antigas que vieram alterar e transformar o modo como conhecemos o passado. Simulações em tempo real contêm, por vezes personagens virtuais sem individualidade e vagueando num ambiente, sem um objetivo específico. Essa situação é particularmente problemática no campo do património cultural em simulações do passado em que o realismo individual é critico.

Baseada na ideia de recriação de uma visão do passado através de imagens, de um tempo e espaço que já não existem, a realidade virtual foi utilizada como plataforma para materialização dessa ideia. O principal objetivo foi recriar virtualmente ambientes urbanos desaparecidos, para isso foi desenvolvida uma ferramenta que automatiza a inserção de personagens virtuais adotando uma variabilidade de comportamentos num cenário urbano 3D previamente criado e que recria uma época histórica.

Este projeto veio dar continuidade a um trabalho prévio que definiu um *pipeline* para a reconstrução digital de espaços urbanos históricos. A solução proposta é de baixo custo e usou-se a cidade de Mértola como caso de estudo. Neste trabalho: i) Procedeu-se à identificação de um processo adequado para expandir o terreno em torno do castelo de Mértola de forma realista; ii) Criou-se uma ferramenta interativa que permite inserir animais e personagens virtuais inteligentes dentro do cenário produzido no passo anterior.

Palavras-chave: Realidade Virtual, Humanos Virtuais, Ambientes Virtuais, Arqueologia Virtual, Herança Cultural, Mértola Virtual.

Abstract

In recent decades, technology has allowed us to create virtual worlds, recreations of ancient civilizations that have changed and transformed the way we know the past. Real-time simulations sometimes contain virtual characters without individuality and wandering in an environment without a specific purpose. This situation is particularly problematic in the field of cultural heritage in past simulations where individual realism is critical.

Based on the idea of recreating a vision of the past through images, a time and space that no longer exist, virtual reality was used as a platform to materialize this idea. The main objective was to virtually recreate missing urban environments, for this purpose a tool was developed that automates the insertion of virtual characters by adopting a variability of behaviors in a previously created 3D urban scenario that recreates a historical epoch.

This project continued a previous work that defined a pipeline for the digital reconstruction of historic urban spaces. The proposed solution is low cost and the city of Mértola was used as a case study. In this work: i) An appropriate process was identified to expand the terrain around Mértola castle in a realistic way; ii) An interactive tool was created that allows to insert animals and intelligent virtual characters into the scenario produced in the previous step.

Keywords: Virtual Reality, Virtual Humans, Virtual Environments, Virtual Archeology, Cultural Heritage, Virtual Mértola.

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Siglas

HCD - Herança Cultura Digital.
HMD - Head Mounted Display.
FBX – Filmbox.
OBJ – Wavefront OBJ.
2D – Bidimensional.
3D - Tridimensional.
HV- Humanos Virtuais.
HVi – Herança Virtual.
HC – Herança Cultural.

Capítulo 1

Introdução

Representações da realidade ou da imaginação sempre fizeram parte da vida do ser humano permitindo-o expressar-se ao longo do tempo, desde desenhos primitivos, figuras e pinturas até o cinema, passando por jogos, teatro, ópera, e outras expressões artísticas. O uso do computador potencializou e convergiu tais formas de expressão, viabilizando a multimédia, que envolve textos, imagens, sons, vídeos e animações. Ao mesmo tempo, os videojogos ganharam um espaço extraordinário, explorando a interação. Não demorou para que todas essas tecnologias convergissem e, rompendo a barreira do ecrã, passassem a gerar ambientes tridimensionais interativos em tempo real, através da realidade virtual.

A Realidade Virtual (RV) é uma tecnologia que tem vindo a ganhar terreno em diversas áreas, e tem vindo a demonstrar ser capaz de apresentar algumas soluções para problemas comuns, o seu conceito não é totalmente recente, tendo passado já alguns anos desde as primeiras referências a este tipo de tecnologia informática [1].

O projeto descrito na presente tese tem a RV como premissa no desenvolvimento de uma ferramenta útil para contextos no âmbito da reconstrução virtual de ambientes históricos urbanos desaparecidos, com personagens virtuais adotando diversos comportamentos inseridos em cenários tridimensionais interativos. A utilização desta reconstrução reveste-se de inegável valor educativo, pois permite apreciar com detalhe o espaço urbano da época e o tipo de arquitetura utilizada pela civilização naquela altura, assim como levar a uma melhor compreensão do estilo de vida e da herança cultural deixada.

Este trabalho enquadra-se na unidade curricular Dissertação/Projeto de Engenharia Informática (DPEI) que está inserido no 2º ano do Mestrado em Engenharia Informática (MEI) da Faculdade de Ciências da Universidade de Lisboa (FCUL), na unidade de I&D Biosystems and Integrative Sciences Institute (BioISI), em particular no grupo, MAS (Agent and Systems Modeling).

1.1 Motivação

Para um vasto número de pessoas, o conceito de Ambiente Virtual está, de facto, intimamente ligado à sua aplicação em diferentes áreas. No entanto, recentemente surgiu uma nova área, que usa as diferentes aplicações dos Ambientes Virtuais. Esta nova área dá pelo nome de Herança Cultural Virtual e tem vindo a tornar-se cada vez mais importante, no que diz respeito à reconstrução, conservação, preservação e interpretação do imenso legado histórico da Humanidade [2].

De facto, a vasta área de aplicação dos ambientes virtuais permite-nos concluir que estes constituem uma forma privilegiada de divulgação de aspetos culturais e científicos. A Cultura e

a Ciência são assimiladas e compreendidas mais facilmente, uma vez que os utilizadores podem interagir com o mundo virtual e manipular os dados, produzindo diferentes modelos de conhecimento.

Devido ao facto de Portugal ser um país com uma longa história e um extenso e rico património arquitetónico e cultural que remonta ao período da ocupação romana, árabe e com muitos exemplos anteriores, faz com que a reconstrução de ambientes históricos, em particular os que já não existem e, por isso, não podem ser visitados e apreciados pessoalmente seja atualmente um ramo próspero e ativo dentro do campo da educação, arquitetura e história.

Todos nós já nos deparamos em diversos locais com elementos históricos e artísticos que o tempo degradou, levando-nos a imaginar como estes seriam quando enquadrados nos seus momentos áureos. Com esta visão a motivação para este trabalho é contribuir para a definição de uma solução de baixo custo para a reconstrução de espaços históricos urbanos que permitam a inserção de personagens virtuais interativos que recriem hábitos da época, simulando assim um cenário virtual urbano interativo, que genericamente todos pudessem controlar e do qual pudessem usufruir seja em museus ou escolas, proporcionando desta forma uma viagem no tempo e a possibilidade de devolver à memória coletiva um passado extinto, através da recriação e simulação de ambientes que já não fazem parte das nossas vidas.

1.2 Objetivos

O objetivo fundamental delineado para este trabalho é introduzir processos que garantam variabilidade de comportamentos dos personagens virtuais que se movimentam em cenários tridimensionais. Pretende-se conceber e concretizar uma ferramenta útil no contexto da herança cultural digital (recriação de civilizações antigas).

A melhoria do realismo em cenas virtuais 3D é um desafio contínuo de pesquisa, tanto em termos de renderização quanto de comportamento exibido pelos personagens nos mundos virtuais. No entanto, as simulações em tempo real são, com muita frequência, limitadas a personagens sem individualidade e vagueando em um ambiente sem um objetivo específico. Essa situação é particularmente problemática no campo do património cultural em simulações do passado em que o realismo do comportamento individual é crítico, com isso um dos objetivos é atenuar esta problemática introduzindo variabilidade e espontaneidade no comportamento da animação dos personagens virtuais.

Este projeto usa uma recriação prévia da muralha e casas de Mértola na idade Média e desenvolve uma aplicação com um completo sistema de avatar característicos da época, personagens não-jogadores com inteligência artificial, vegetação apropriada. A interface foi pensada de maneira agnóstica, isto é, que se adapta a todo tipo de plataforma, desde que seja no âmbito da ferramenta Unity 3D, dando ao utilizador o poder de decisão e flexibilidade, oferece também oportunidades de aprendizagem, onde o utilizador poderá entrar no mundo selecionando o número avatares, assim como o seu género, o ponto de aparecimento para a navegação, e também permite que os utilizadores mudem de participantes para espectadores dentro da simulação, de forma a mergulhar em todos os aspetos da vila simulada.

Tendo em conta a preocupação de obter uma solução de baixo custo é essencial recorrer sempre que possível a ferramentas gratuitas ou com licenças académicas e que permitam fluxos de dados entre si, para isso é necessário identificar estas ferramentas, testando a integração de modelos tridimensionais criados nas distintas ferramentas.

1.3 Contribuições

A proposta aqui apresentada pode contribuir em duas vertentes. Primeiro enquanto material tecnológico de apoio à divulgação e uso das tecnologias gráficas tridimensionais, e segundo como contributo histórico no estudo, promoção e conhecimento de ambientes históricos antigos.

No decurso deste trabalho concebeu-se e implementou-se uma solução que recria, uma simulação num cenário urbano com humanos e animais virtuais adotando uma variabilidade de comportamentos que recria uma época histórica, o caso de estudo específico ao qual foi aplicada a solução é a vila de Mértola, uma vila museu localizada em Portugal, na região do Baixo Alentejo, durante o seu período de ocupação Islâmica (Séc. VII a Séc. XIII).

Para geração desta ferramenta, o projeto foi divido em 2 fases. A primeira fase consistiu na identificação dos processos para a conversão e expansão do terreno gerado pelo pipeline descrito na tese do Alexandre Carvalho [3].

Um aspeto tido em consideração foi que o terreno do modelo fosse alargado mantendo o realismo, tanto quanto possível próximo do terreno original.

Na segunda fase do projeto inseriram-se personagens virtuais, cujo comportamento se baseia na execução de *scripts* de Inteligência Artificial. Criou-se uma interface da aplicação, ela foi pensada para ser utilizada em diferentes projetos dentro do âmbito do Unity3D, oferecendo oportunidades de aprendizagem no estudo histórico e conhecimento de ambientes antigos, proporcionando ao utilizador uma imersão virtual da cultura, costumes e comportamento da época. O sistema de avatar permite que os visitantes entrem no mundo selecionando classe (compradores e vendedores num mercado) e o género, assim como o número de personagens virtuais previamente animados com suas respetivas máquinas de estados e *scripts* de inteligência artificial e a localização dentro da vila de Mértola onde começará a simulação.

A aplicação desenvolvida recorreu a diversas ferramentas de software. No decorrer do projeto de investigação foram escritos dois artigos científicos com contributos substanciais do trabalho descrito nesta tese.

O primeiro artigo apresentado na conferência internacional, ARQUEOLÓGICA 2.0 - 8th International Congress on Archaeology, Computer Graphics, Cultural Heritage and Innovation na cidade de Valência (Espanha):

- *A. P. Cláudio, M. B. Carmo, A. A. Carvalho, W. Xavier, R. F. Antunes, "Virtual cities inhabited by autonomous characters: a pipeline for their production". Presented at 8th International Congress on Archaeology, Computer Graphics, Cultural Heritage and Innovation 'ARQUEOLÓGICA 2.0' in Valencia (Spain), Sept. 5 – 7, 2016.*

O segundo artigo trata-se de um texto mais alargado, dos mesmos autores, publicado na revista científica, "Virtual Archaeology Review", depois de revisto:

- *A. P. Cláudio, M. B. Carmo, A. A. Carvalho, W. Xavier, R. F. Antunes, "Recreating a medieval urban scene with virtual intelligent characters: steps to create the complete scenario."* ***Virtual Archaeology Review****, [S.l.], v. 8, n. 17, p. 31-41, july 2017. ISSN 1989-9947.*

1.4 Estrutura do documento

Este documento está organizado da seguinte forma:

- **Capítulo 2** – Conceitos e trabalho relacionado

Neste capítulo, são apresentados conceitos base e descritos diversos projetos e estudos realizados que, de alguma forma, se enquadram no âmbito do desenvolvimento deste projeto.

- **Capítulo 3** – Análise de ferramentas e caso de estudo
  Neste capítulo descreve-se o trabalho realizado referente ao estudo de ferramentas de software considerados adequado ao contexto do projeto, os recursos envolvidos no desenvolvimento do processo de criação do cenário de Mértola Virtual, e os requisitos mínimos para a execução da mesma. Introduz-se também caso de estudo para o cenário de Mértola Virtual.
- **Capítulo 4** – Ferramenta Mértola virtual
  Neste capítulo, é descrita a implementação da ferramenta Mértola Virtual, assim como o conjunto de passos para a realização da mesma.
- **Capítulo 5** – Conclusões e trabalho futuro
  Neste capítulo, é apresentada uma conclusão final do trabalho assim como sugestões para o desenvolvimento de trabalho futuro com base nas etapas alcançadas.

Capítulo 2

Conceitos base e trabalhos relacionados

Neste capítulo, serão descritos alguns conceitos e trabalhos que são relevantes no âmbito deste projeto.

2.1 Realidade Virtual

A Realidade Virtual (RV) é uma tecnologia que permite criar ambientes simulados por computador que recriam ambientes reais ou imaginários. A realidade virtual recria mundos fictícios onde o utilizador deve idealmente sentir-se presente e com o qual consegue interagir [4]. O termo "Realidade virtual" foi inicialmente apresentado por Jaron Laniner em 1989, que descreveu a tecnologia de simulação por computador. Após décadas de desenvolvimento, a tecnologia de realidade virtual tem-se estendido nas mais variadas áreas, como por exemplo, a Visualização Científica em Medicina, a Educação, o Entretenimento, a Arquitetura, a Arqueologia, a Herança cultural digital [5]. Existem muitas outras aplicações envolvendo treino, simulação, cidades virtuais. A realidade virtual vem proporcionando uma nova maneira de ver as coisas conhecidas ou o desenvolvimento de novas aplicações [6].

Embora seja imenso o potencial de aplicações da realidade virtual nas mais diversas áreas a sua aceitação popular, existe uma variedade de entendimentos e interpretações nas funções e implicações dos sistemas de realidade virtual. Como resultado, existem diferentes significados e definições do termo realidade virtual [5].

Uma definição, que se enquadra parcialmente no âmbito desse projeto, caracteriza a Realidade virtual como uma interface avançada para aplicações computacionais, que possibilita ao utilizador a movimentação (navegação) e interação em tempo real, em um ambiente tridimensional podendo fazer uso de dispositivos multissensoriais, para atuação ou feedback [6]. Ao interagir com um ambiente simulado através de Realidade Virtual, o utilizador pode experimentar uma Sensação de presença (*Sense of Presence*), como se estivesse realmente no ambiente virtual. Esta sensação é propiciada no utilizador através das 3 principais variantes que estão associados à RV: imersão, interação e imaginação [7]. Esta triangulação de conceitos é conhecida como os 3 I's da RV (Figura 2.1).

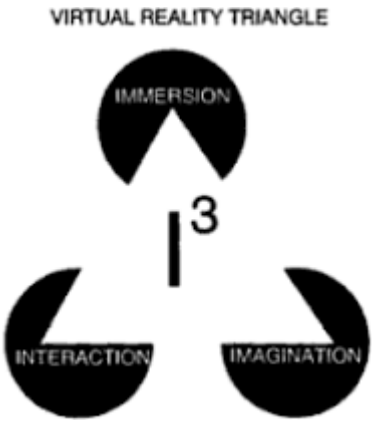


Figura 2.1: Triângulo da RV [7]

Imersão

A imersão é a capacidade que o sistema tem de enganar o utilizador, fazendo com que este se sinta presente no ambiente virtual, ou seja, é a perceção de se estar fisicamente presente em um mundo não físico. Quanto maior for a capacidade do nível de isolamento relativamente ao ambiente físico em que o utilizador se encontra, maior será o nível de imersão, com isso o utilizador terá perceção de fazer parte do ambiente virtual [8].

Existem três níveis principais de imersão no uso de RV:

I. **Não-imersivo**, refere-se à experiência obtida, por exemplo, através de um computador de secretária, onde o ambiente virtual é gerado sem a necessidade de utilizar nenhum equipamento imersivo específico. A ilusão de estar imerso é gerada por respostas dos personagens gerados pelo computador e ações que podem ser tomadas pelo utilizador [4].

II. **Semi-imersivo**, trata-se de uma experiência mais avançada relativa a experiência não-imersiva, isto é, neste nível de imersão é feito o rastreamento do movimento da cabeça do utilizador e, de acordo com este movimento a imagem gerada pelo ambiente virtual é modificada, passando a seguir este movimento, portanto melhoram a sensação de "de lá estar", para este sistema normalmente é utilizado um monitor convencional (muitas vezes com óculos de obturador LCD mais conhecidos como *shutter glasses* para visualização estereoscópica), entretanto, geralmente não suportam a entrada sensorial [9].

III. **Totalmente imersivo**, neste tipo de experiência, o utilizador é exposto a uma imersão num mundo gerado por computador com a ajuda de HMD (*head-mounted-display*), onde visualiza a cena de acordo com a sua orientação e posição, essa experiência pode ser melhorada pela interface auditiva, háptica e sensorial [4].

Interatividade

Está associada à capacidade do sistema de Realidade Virtual de detetar e interpretar as ações do utilizador e modificar em tempo real o mundo virtual e as ações sobre ele, isto é, o sistema tem que reagir as ações do utilizador para que experiência num sistema de RV pareça autêntica.

Imaginação

A imaginação diz respeito à capacidade de o utilizador percecionar o ambiente virtual como real. Esta capacidade é uma característica individual e depende da imaginação do utilizador. No entanto, e apesar de depender da capacidade de abstração do utilizador, é um facto de grande importância para quem desenvolve a aplicação de RV já que a classificação do sistema por parte de um utilizador depende da sua capacidade imaginativa e de este ser capaz de atenuar as falhas do mundo virtual, imaginando-o como real.

2.1.1 Contexto histórico

O desenvolvimento tecnológico da realidade virtual teve o seu início na década de 1950 e tem crescido consideravelmente nos últimos anos. A Realidade Virtual começou na indústria de simulação, com os simuladores de voo que a força aérea do Estados Unidos passou a construir logo após a Segunda Guerra Mundial [10].

A indústria de entretenimento também teve um papel importante, ao construir o primeiro sistema de Realidade Virtual simulador nos anos 1960-1962. Morton Heilig criou um simulador

multissensorial designado por “Sensorama”, (Figura 2.2) era uma espécie de cabine que permitia ao utilizador expor-se a uma combinação de estímulos, onde um filme pré-gravado a cores era passado juntamente com som estéreo, aromas, vento e experiências de vibração, tudo isso proporcionando ao utilizador a participação em uma viagem multissensorial, esta foi a primeira abordagem para criar um sistema de realidade virtual e tinha todas as características de tal ambiente, mas não era interativo [9] [14].

Embora não tenha sido um sucesso comercial, o “Sensorama” era uma invenção curiosa para sua época, podendo ser considerado como a primeira experiência em 4D da história, e as unidades que existem ainda funcionavam recentemente [11]. O invento patenteado por Morton Heilig em 1962 já utilizava um dispositivo para visão estereoscópica, além de ter sido o precursor da imersão do utilizador num ambiente artificial.

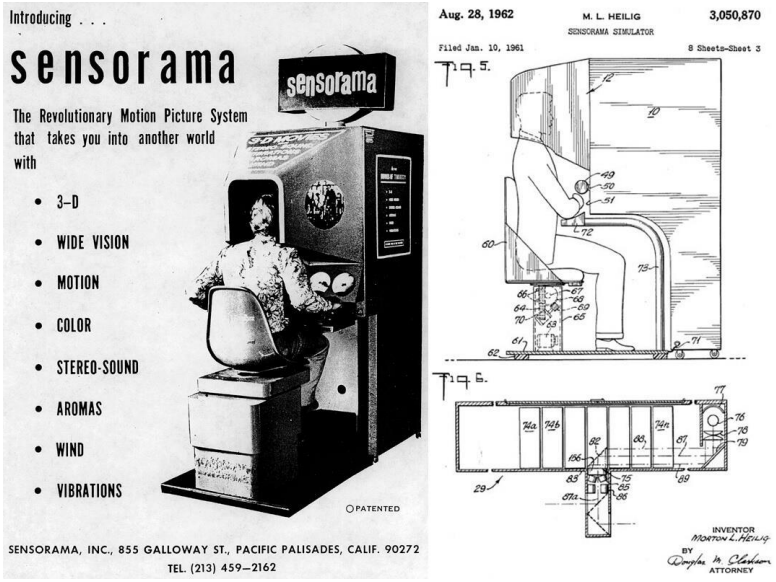


Figura 2.2: Sensorama, o primeiro dispositivo de Realidade Virtual [12]

A partir do Sensorama, seguiram-se décadas de pesquisa e desenvolvimento durante as quais diversos dispositivos de RV foram criados e aprimorados, sempre buscando trazer experiências imersivas ao utilizador. Na Universidade de Utah, em 1965, Ivan Sutherland apresentou a sua tese de doutoramento em que desenvolveu uma caneta ótica para desenhar diretamente no ecrã de um computador dando o primeiro passo para a criação da computação gráfica interativa e para a criação da realidade virtual [10]. Em1968, temos o que foi considerado o primeiro o *Head-Mounted Display* (HMD) a ser criado [9]. O aparelho, que levou o nome de a Espada de Dâmocles (*Sword of Damocles*), construído por Ivan Sutherland, juntamente com o Bob Sproull, era tão grande e pesado, que tinha que estar preso ao teto. Era bastante primitivo no que se refere a interface e gráficos (que eram em *wire-frame*), mas foi um grande passo na direção da tecnologia que é utilizada hoje [11].

Nessa época, Myron Krueger, na Universidade de Wisconsin, iniciou estudos com realidade artificial e, em 1975, desenvolveu o *Videoplace*, onde uma câmara de vídeo capturava a imagem dos participantes e projetava-a em 2D num grande ecrã. Os participantes podiam interagir uns com os outros e com objetos projetados nesse ecrã, graças às técnicas de processamento de imagens para determinar as suas posições no espaço do ecrã 2D. Essa técnica tornou-se conhecida como Realidade Virtual de Projeção [9] [13].

No entanto em 1982, Thomas Furness desenvolveu no laboratório de pesquisas médicas da Força Aérea Americana o *Visually Coupled Airborne Systems Simulator* (*VCASS)*, conhecido

como “Super Cockpit”. Era um simulador que utilizava computadores e vídeo-capacetes interligados com áudio e vídeo e conseguia representar o espaço 3D de uma cabine de avião, permitindo assim o aprendizado e treino de voo do piloto com um grande grau de liberdade para a movimentação [10].

A partir da década de 80 a NASA demonstrou interesse em criar um sistema de Realidade Virtual para usar como ferramenta de treino, foi então que em 1984, como consequência de uma nova tecnologia de visores de cristal líquido (LCD), Michael McGreevy, começou a trabalhar no projeto *Virtual Environment Display* (*VIVED*). Numa máscara de mergulho foram colocados componentes de áudio e vídeo usando dois visores de cristal líquido com visualização de imagens estereoscópicas e com pequenos altifalantes acoplados. No ano seguinte em 1985, Scott Fisher juntou-se a esse projeto com o objetivo de incluir nele: luvas de dados, reconhecimento de voz, síntese de som 3D, e dispositivos de resposta (*feedback)* tátil [14].

Exatamente em 1986 a NASA já detinha um ambiente virtual que permitia aos utilizadores manipular objetos virtuais através do movimento das mãos utilizando uma luva especial chamada “*DataGlove*”, desenvolvida com sensores de fibra ótica [10] [14]. A partir da premissa de que os projetos da NASA poderiam tornar-se equipamentos comercializáveis, empresas de software e grandes corporações de informática começaram a desenvolver e vender serviços e produtos voltados para RV, foi então que em 1989, a *AutoDesk*, apresentou o primeiro sistema de Realidade Virtual voltado para computadores pessoais [10]. Dessa forma, a realidade virtual consolidou-se e evoluiu, tornando-se uma indústria, e os sistemas evoluíram até aos dispositivos imersivos atuais como por exemplo o *“Oculus Rift”*[1].

2.1.2 Áreas de aplicação

A realidade virtual tem aplicação numa vasta gama de áreas de investigação, que tiram partido das suas possibilidades de maneiras diferentes e com propósitos distintos. Se esta começou por ser encarada como algo utópico saído de livros e filmes de ficção científica, a sua utilização foi ganhando interesse de investigadores de várias áreas, e a sua aplicação real foi saindo do domínio da imaginação e ganhando importância, à medida que as tecnologias iam evoluindo e amadurecendo e permitiam aplicações cada vez mais complexas e menos lúdicas. Dessa forma, expectativas foram criadas em torno dos sistemas de RV e despertaram o interesse de diversas empresas de diferentes setores. Atualmente, devido à procura e capacidade criativa das pessoas através da RV, novas aplicações são criadas nas mais variadas áreas do conhecimento e de maneira bastante diversificada. De seguida, enumeram-se alguns exemplos ilustrativos de aplicações de RV em diferentes contextos.

Medicina e Terapêutica

A medicina e as áreas de saúde em geral têm-se beneficiado dos avanços tecnológicos apresentados pela RV, nos últimos anos, com isso tem sido uma das áreas em que mais se investiu esforço de desenvolvimento. Os investigadores acreditam que a RV proporciona um importante recurso para o ensino e treino em estruturas anatómicas e na simulação de operações, isto é, o aluno pode explorar continuamente as estruturas de interesse, podendo separá-las ou agrupá-las com as mais diferentes formas de imersão, visualização e exploração.

A Realidade Virtual também tem sido utilizada em treino de procedimentos cirúrgicos. A vantagem proporcionada pelo uso de um HMD, é que o corpo do paciente pode ser manipulado

[1] Oculus Rift: https://www.oculus.com/rift/

para ser translúcido ou transparente, expondo claramente as zonas cruciais. Isso torna muito mais fácil para o cirurgião treinar as operações nas áreas corretas do corpo, tornando a vida do paciente consideravelmente mais segura [4].

A maravilha da RV é que ela pode até minorar limitações físicas. Os tetraplégicos utilizam a tecnologia de RV para mover objetos no ecrã do computador usando apenas seus olhos ou expressões faciais [4].

Já na área da terapêutica a sua utilização é mais voltada para o paciente, com vista a tratamento de fobias e Stress Pós-Traumático por exemplo. Os ambientes virtuais são constituídos essencialmente por imagem e som, com modelos simples, e mostraram-se bastante eficazes no tratamento de Zoofobia (medo irracional de animais) ou Acrofobia (Medo de Alturas), constituindo um meio intermédio entre os simulacros e a exposição às situações reais. A marinha dos E.U.A. recorre a este método para tratamentos de Stress Pós-Traumático de veteranos, especialmente da guerra no Iraque (Figura 2.3).

Figura 2.3: Utilização da RV pela marinha americana [17].

Entretenimento

A indústria do entretenimento é uma utilizadora entusiasta da RV, mais notavelmente em jogos, trata-se de uma área com enorme dimensão e muito investimento financeiro. A indústria de videojogos supera um grande impacto no campo da RV, devido ao público dessa indústria ser enorme. Em função disso, apostas promissoras marcaram o início de um novo percurso em toda a indústria, essa aposta refere-se aos óculos de RV, atualmente existem no mercado vários dispositivos *HMD* entre os quais podemos destacar: *Sony Playstation VR*[2], *HTC Vive*[3], *Oculus Rift da Oculus VR, Samsung Gear VR*[4], *Google Daydream View*[5].

Arquitetura

A realidade virtual tem-se mostrado uma ferramenta promissora na engenharia e na arquitetura. Antigamente, tanto arquitetos como os próprios clientes do projeto, para ter uma perspetiva visual do projeto, por exemplo de um edifício tinham de recorrer a *blueprints*, porque, não existia uma tecnologia suficiente que facilitasse a visualização preliminar do trabalho do

[2] Sony Playstation VR: https://www.playstation.com/pt-pt/explore/playstation-vr/

[3] HTC Vive: https://www.vive.com/us/product/vive-virtual-reality-system/

[4] Samsung Gear VR: https://www.samsung.com/global/galaxy/gear-vr/

[5] Google Daydream View: https://vr.google.com/daydream/smartphonevr/

arquiteto. Esse facto acabava por tornar o processo muito mais arriscado, difícil de percepcionar e consequentemente mais custoso.

Sendo assim, a RV permite que o arquiteto tenha liberdade para projetar e explorar protótipos estruturais, além de permitir uma visualização prévia bastante precisa. Desta forma é possível fazer modificações no projeto antes da construção atual começar, consequentemente economiza-se tempo e custos.

Um exemplo destacável e recente de aplicação da RV na arquitetura, surgiu através da lacuna que existe entre projetos em formato 3D e em RV, mais precisamente no auxilio da conversão de plantas que estão no modelo tradicional, ou seja se um utilizador que já dispõe de vários projetos que estão a decorrer, porém deseja testá-los recorrendo a RV, é possível testar a tecnologia sem perder todo o trabalho que já foi realizado através de um software desenvolvido pela *startup* americana *IrisVR* chamado *Prospect*[6], ele é capaz de converter projetos, originalmente em formato 3D, em realidade virtual, o programa é compatível por exemplo com os *softwares*: *Revit, Sketchup, Rhino* e ficheiros com extensão .OBJ, estes ficheiros são convertidos em experiências de realidade virtual navegáveis para *o HTC Vive e Oculus Rift*. O programa também permite verificar se os recursos de realidade virtual na arquitetura realmente atendem às suas necessidades do utilizador.

Treino e Educação

Mudanças drásticas nos métodos de treino e educação têm sido observados com a introdução da RV. O conteúdo entregue é muito mais rico e robusto através da estimulação de mais sentidos, torna a informação mais fácil de compreender. Na área de treino, a RV está sendo utilizada para reduzir custos, recriando ambientes virtuais e simulando cenários da vida real [15]. A indústria militar, tem há anos profundo interesse nessa área em decorrência de vários fatores. O primeiro, e mais óbvio, é a segurança de treinar um militar em ambiente simulado e controlado, por um lado considerando a vida do piloto, o custo do equipamento, que geralmente chega a ter um elevado custo. A Boeing por exemplo, recorre a simuladores para o treino de novos pilotos, para os preparar para situações adversas e perigosas recriando virtualmente esse tipo de situação, ou para testar um novo modelo de aeronave (Figura 2.4).

Figura 2.4: Simulador de treino para pilotos da Boeing interior (esquerda), e exterior (direita) [18]

6 Prospect: https://irisvr.com/prospect

Já na educação, estudos demonstram que a utilização de RV aumenta a motivação dos alunos e melhora o seu desempenho académico [16]. A utilização deste género de tecnologia estimula os alunos a ter um papel mais dinâmico na sua aprendizagem uma vez que muitas aplicações permitem uma exploração autónoma do mundo virtual. Entretanto, para professores a utilização do recurso à RV é bastante diversificado e robusto, podendo por exemplo, levar os seus alunos a "entrar" dentro do corpo humano e observar o funcionamento do mesmo, desde órgãos a células. As aplicações *InCell*[7], *InMind*[8], *Anatomyou VR*[9] são exemplos disso, já a aplicação *Titans of Space*[10], permite que o professor possa levar toda a turma a uma viagem espacial nos possíveis planetas em estudo, promovendo um estudo provavelmente mais interessante e detalhado sobre determinados aspetos dos assuntos que leciona.

Arqueologia e Herança cultural

A crescente preocupação com a preservação, interpretação e divulgação do Património Arqueológico despontou o interesse em unir novas tecnologias, entre elas a RV, no âmbito da representação científica e modelação geométrica de artefactos arqueológicos como também na representação de sítios arqueológicos.

A imersão num mundo arqueológico virtual transmite ao utilizador um conjunto de sensações, permitindo percorrer, ruas, entrar nos edifícios, analisar detalhes, enfim visitar todo um espaço antigo livremente e que pode ser enriquecido com outras informações importantes. De igual modo constrói-se uma interação que promove a ideia de realidade, ao tocar com os objetos, subir escadas, olhar pelas janelas. Algumas experiências neste campo vão sendo produzidas atualmente por todo o mundo [19].

No âmbito da herança cultural, com a RV é possível regressar ao passado e visitar virtualmente vários locais de interesse histórico, visualizar exemplos da arquitetura antiga, túmulos, palácios, castelos, torres, cidades antigas e museus. Este tipo de tecnologia aplicado a museus, não só torna mais proveitosa, em termos de conteúdo, para o adulto, como torna a exposição mais divertida e estimulante também para crianças, assumindo a realidade como menos abstrata e mais compreensível. Aplicações como: *Sites VR*[11], *Egypt VR*[12], permitem que o utilizador possa entrar, explorar e analisar os locais ao pormenor.

Nas secções seguintes descreve-se brevemente algumas abordagens de Humanos Virtuais, Herança Cultural e Herança Virtual.

2.2 Humanos virtuais

Os Humanos Virtuais (HV) são habitualmente modelos 3D do corpo humano, que podem possuir animações faciais e corporais permitindo simular o comportamento humano desejavelmente, de forma realista.

Segundo Tori [6], um HV é uma representação gráfica de uma pessoa real no ambiente virtual, conjuntamente com seu comportamento. Já Torres [21] refere que os HV são modelos computacionais de pessoas, utilizados tanto como substitutos para pessoas reais em avaliações

[7] InCell: https://luden.io/incell/

[8] InMind: https://luden.io/inmind/

[9] Anatomyou VR: https://anatomyou.com/en/

[10] Titans of Space: http://titansofspacevr.com/

[11] Sites VR: http://www.sitesinvr.com/

[12] Egypt VR: https://play.google.com/store/apps/details?id=com.amr.egypt.vr360&hl=en_US

ergonómicas de projetos baseados em computador, em que os HVs podem ter de apresentar uma aparência bastante realista, coerente com princípios fisiológicos e biomecânicos. Restrições funcionais podem ser aplicadas a estes modelos de forma que os seus movimentos satisfaçam as limitações humanas. Além disso, ao invés de ficarem restritos a movimentos previamente definidos, os humanos virtuais são capazes de comportar-se de forma autónoma e inteligente em ambientes dinâmicos, podendo apresentar até mesmo individualidade e personalidade.

Um dos primeiros modelos de HV surgiu em 1959. Foi desenvolvido para a Boeing, com intuito de ser utilizado no estudo e análise ergonómica de aviões em simuladores, como resultado, surgiram algumas das primeiras aplicações em computação gráfica para modelar uma figura humana e o seu movimento. Com essa base, vários modelos foram desenvolvidos, porém estes modelos eram básicos e pouco articulados.

Em 1971, Parke produz uma representação da cabeça e face na Universidade de *Utah*, e três anos depois avança em modelos paramétricos suficientemente bons para produzir um rosto muito mais realista [22].

No início dos anos 80, Tom Calvert utilizou um sistema rudimentar de captura de movimentos, onde potenciómetros eram anexados a um corpo e o *output* gerado, animava as figuras do computador, sendo utilizados para estudos coreográficos e na avaliação clínica do movimento. De alguma forma, os seres humanos virtuais eram funcionais, porém não muito atraentes. A partir desta altura, várias empresas e grupos de investigadores produziram curtas-metragens realizadas com HV. Em 1985, o filme *"Tony de Peltrie",* utiliza pela primeira vez técnicas de animação facial para contar uma história, desde então, se começa a usar animação facial em filmes e nesta mesma década, pela primeira vez, um filme ganha o Oscar na categoria *Best Animated Film*, em 1988.

Na década de 1990, a ênfase mudou para a animação em tempo real e a interação com os ambientes virtuais, com personagens capazes de conversar e fazer gestos. Nesta altura os HV começam a ser capazes de interagir autonomamente com o ambiente virtual com uma certa inteligência [22].

2.3 Herança Cultural

A maioria dos pensadores contemporâneos concorda que estamos passando por um período de mudança histórica, construindo um conceito diferente e um novo modelo de inter-relação social [23]. Com essa mudança, é imprescindível garantir a preservação cultural de um povo através da Herança Cultural (HC), que na sua definição pode ser considerada como um conjunto de valores transmitidos, através do processo de socialização entre gerações, e que são característicos da identidade de um povo que de um modo dinâmico vai sendo criada por uma interação de diferentes elementos geográficos, ambientais, sociais e outros fatores externos e que, de uma perspetiva antropológica, irá afetar os modos de ser, pensar e agir de uma pessoa. A Herança Cultural é como a transição de uma cultura entre povos, de uma geração às seguintes. Uma cultura viva e formativa deve estar aberta para o futuro, mas ancorada no passado [3]. Portanto, é imprescindível garantir a sua preservação através do desenvolvimento de metodologias. Dessa forma, a Herança cultural estará salvaguardada para as gerações futuras.

No decorrer do século XX com o advento da era digital, as fronteiras da interação humana com a cultura foram afastadas. As técnicas de uma infinidade de campos estão a ser constantemente desenvolvidas e refinadas, levando a essa expansão. Estruturas de dados detalhadas permitem aos investigadores explorar novas ligações entre exposições, técnicas de modelação que permitem conservacionistas explorar todos os tipos de artefactos desde línguas extintas até sítios arqueológicos distantes e aplicações inovadoras que encantam e inspiram

visitantes de um museu por exemplo. Todos esses desenvolvimentos interessantes estão sob a égide da Herança Cultural Digital. Este é um campo amplo com muitos subcampos e áreas ativas de pesquisa, mas sua principal preocupação é o uso da média digital na preservação e exploração da Herança Cultural [24]. Portanto, a documentação, interpretação, restauro e disseminação são consideradas tarefas cruciais.

Nos últimos anos, a UNESCO privilegiou o conceito de "Herança integral". Essa compreensão do património como realidade integral requer uma perceção coletiva e o estabelecimento de redes culturais que ampliem a perceção, combinando monumentos, patrimônios, museus, paisagens naturais, paisagens culturais, realidades etnográficas, sítios históricos, memória coletiva e rotas culturais. Nesse objetivo, as novas tecnologias podem se tornar um aliado importante e eficaz [23].

Essas tecnologias permitem produzir cenários de herança cultural digital com objetivo de preservar a memória do passado e realizar a transição para o presente e, possivelmente, direcioná-la para o futuro, onde o maior desafio é desenvolver aplicações que garantam, por um lado, o resgate de património histórico da Humanidade e, por outro lado, proporcionem uma boa usabilidade por parte das pessoas envolvidas no processo.

2.4 Herança Virtual

Os ambientes virtuais inseridos na herança cultural e representados através da média digital são frequentemente classificados como "Herança Virtual" (HVi). Nos dias atuais, as novas ferramentas digitais oferecem-nos à possibilidade de experimentar locais históricos virtualmente reconstruídos, sejam como visitantes, viajantes ou residentes. Embora a HC represente um grande potencial para reconstruir à nossa herança e memória do passado, geralmente o custo é elevado, o desenvolvimento é complexo, criando uma certa resistência na ampla disseminação da HVi. Porém a própria UNESCO, organização mundial que avalia e classifica determinado património como "Património da Humanidade", no *Infokit*[13]*(World Heritage Information Kit)* classifica a importância da nossa Herança Cultural:

"Heritage is our legacy from the past, what we live with today and we pass on to future generations. Our cultural and natural heritage are both irreplaceable sources of life and inspiration, our touchstone, our reference point, our identity".

O termo genérico "herança" refere-se ao estudo da atividade humana não apenas através da recuperação de restos mortais, como é o caso da arqueologia, mas também através de tradições, evidências artísticas e culturais e narrativas. Por outro lado, "Herança Virtual" (HVi) é um termo usado para descrever trabalhos que lidam com Realidade Virtual (RV) e Herança Cultural (HC). Em geral, a Herança Virtual e a Herança Cultural têm significados independentes: a HC refere-se a "propriedades e locais com valor arqueológico, estético e histórico" e a HVi refere-se a instâncias dessas propriedades e locais dentro de um domínio tecnológico [25].

A representação de paisagens, objetos ou locais do passado e o processo global de visualização de dados arqueológicos com o uso da tecnologia da RV formam um subdomínio conhecido como Arqueologia Virtual [25].

Segundo Sanchotene, HVi vem da evolução da arqueologia virtual e apresenta-se com a utilização de técnicas baseadas em computador, com a tecnologia de Realidade Virtual, simulando-se uma comunidade antiga com reconstituições virtuais de um habitat ou de edificações representativas do passado relevante da memória e da cultura de um povo [26].

13 UNESCO, "World Heritage Information Kit": https://whc.unesco.org/document/102072

Existem vários cenários em que a HVi pode ser utilizada, desde um contexto educacional que engloba todos os níveis escolares até um nível mais elevado onde pode ser utilizada para disseminar a informação nos campos da história e da arqueologia para pesquisadores, restauradores e arquitetos.

Com o uso da tecnologia de Realidade Virtual, as aplicações em HVi apresentam hoje um crescimento rápido, com potencial expressivo em áreas de arqueologia pré-histórica, na arquitetura de fortificações militares, na arquitetura religiosa, cidades medievais, cidades ou monumentos destruídos em guerras ou pelo tempo. Estes modelos podem ser recriados como modelos virtuais, e ser examinados com possibilidade de acesso ao seu próprio interior.

2.5 Trabalhos relacionados

Nesta secção apresentam-se alguns trabalhos agrupados do seguinte modo: exemplos de simulações históricas e a sua relação com a Realidade Virtual, sem a utilização de humanos virtuais e exemplos de simulações históricas com humanos virtuais, animados com algum tipo de Inteligência Artificial. Este último grupo corresponde a trabalhos mais próximos do desenvolvido nesta tese. Apresentam-se também 2 tabelas, uma na secção 2.5.1 e outra na secção 2.5.2 onde são comparados vários aspetos e características relacionadas com os projetos.

2.5.1 Simulações Históricas

Nesta secção apresentamos, a partir dos estudos realizados uma visão geral dos projetos existentes que relacionam a Herança Cultural com a Realidade Virtual. Alguns destes trabalhos, apesar de não se referirem diretamente à representação de sítios arqueológicos ou cidades históricas recriadas, possuem abordagens similares, como por exemplo os museus virtuais, que possibilitam passeios imaginários a edificações onde podem ser expostos materiais com valores culturais e/ou arqueológicos. Na tabela 1, comparam-se as principais características de um conjunto de trabalhos que relacionam a Herança cultural com a Realidade Virtual em ambientes virtuais.

Museu Arqueológico Virtual de Herculano [20]

No intuito de preservar as ruínas do sítio arqueológico de Herculano foi projetado o Museu Arqueológico Virtual de Herculano[14], que combina a RV com exposições interativas, é possível realizar visitas ao mercado ou a entrada em casas romanas, acompanhadas por sons e cheiros que reproduzem a atmosfera da época. (Figura 2.5).

Já o projeto *Learning Sites*[15] tem como principal objetivo a recriação de edificações históricas utilizando-se de tecnologias como: a realidade virtual, hipertexto e multimédia. A ideia central é prover conteúdo para educação pública e pesquisa académica, contribuindo na preservação de artefactos arqueológicos, criando um repositório arqueológico digital.

[14] O Museu Arqueológico Virtual: http://www.capware.it/

[15] Learning Sites: http://www.learningsites.com

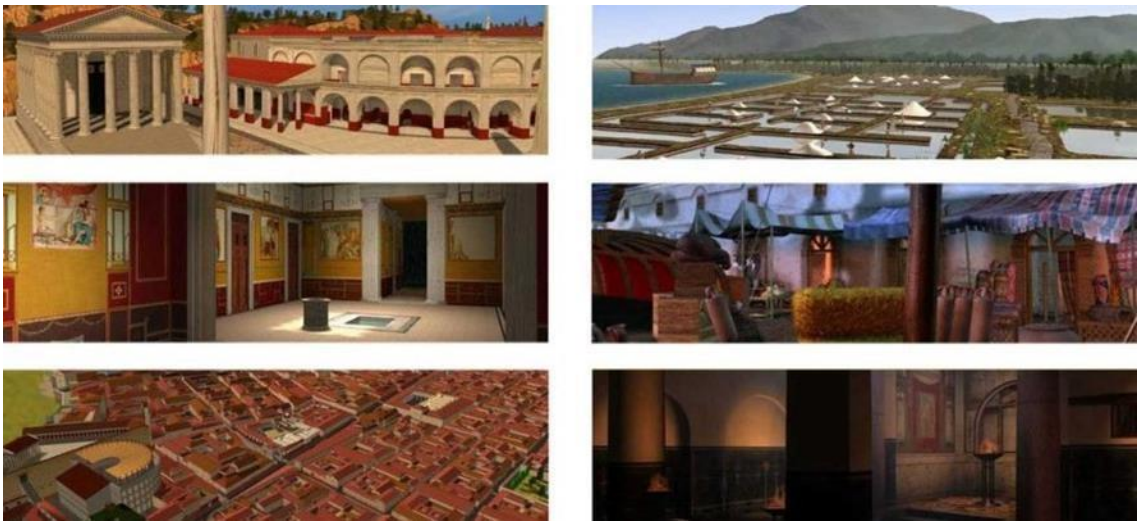


Figura 2.5: Museu Arqueológico Virtual de Herculano [20]

Bracara Augusta [27]

A representação virtual do sítio arqueológico de Bracara Augusta foi realizada a partir de um projeto para avaliar o seu potencial de representação. O objetivo principal do projeto foi a conceptualização de um Sistema de Ambientes Virtuais capaz de representar a cidade romana de Bracara Augusta, desde a sua fundação até ao seu declínio.

Bracara Augusta inclui modelos das termas públicas utilizadas pelos romanos em meados do século XIX. O projeto obedeceu à morfologia do terreno, através da digitalização de curvas de nível obtidas através de um mapa do século XIX. A integração dos modelos tridimensionais consistiu em posicionar as diferentes estruturas e infraestruturas corretamente sobre o modelo do terreno de Bracara Augusta. Todos os modelos do projeto foram convertidos para modelos em formato VRML 2.0 (*Virtual Reality Modeling Language*), que proporciona a visualização em 3D e a interação do utilizador em tempo real.

O realismo dos modelos foi obtido através da definição de materiais, aplicação de texturas e utilização de um modelo de iluminação global. O projeto incluiu ainda a representação dos diversos momentos do local a partir de registos históricos, visto que diversas reformas ocorreram ao longo de sua existência (Figura 2.6).

Figura 2.6: Reconstrução Global de Bracara Augusta [27]

Construções do Império Romano [26]

A Universidade da Califórnia (UCLA) tem vindo a desenvolver no seu laboratório de realidade virtual um conjunto de aplicações interessante para a área. Um de seus projetos concluídos resultou num modelo 3D do Coliseu, onde o utilizador pode realizar uma caminhada virtual pelas galerias e subterrâneos. Conforme apresentada na Figura 2.7, através dessa representação virtual, os engenheiros provaram a historiadores a impossibilidade da evacuação imediata do Coliseu, acreditava-se que em dez minutos 50.000 espectadores podiam deixar a construção. Com o modelo virtual foi mostrado que o projeto não tinha tanta eficiência, pois apresentava corredores estreitos e escuros que dificultariam a saída do público.

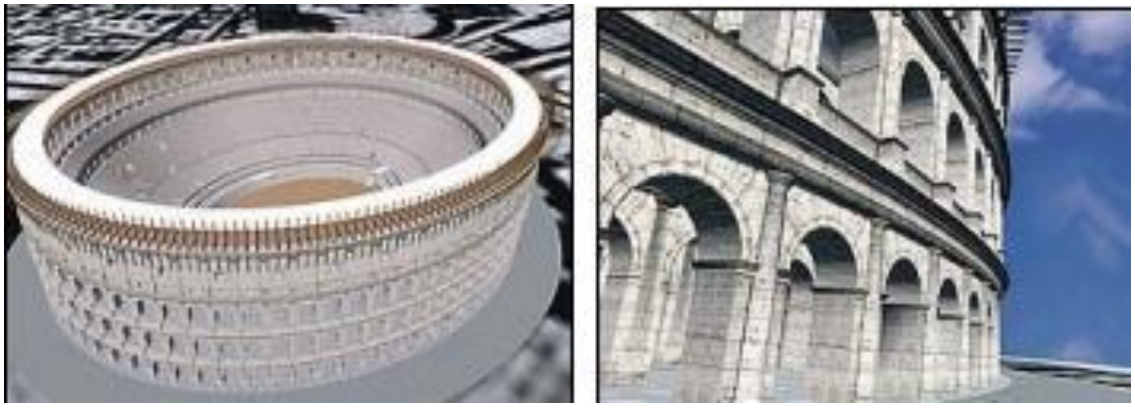


Figura 2.7: Coliseu/ Roma desenvolvido pelo laboratório de Realidade Virtual da UCLA [26]

Reconstrução Virtual da antiga cidade ibérica de Ullastret [28]

Com objetivo de reconstruir virtualmente a cidade de Ullastret, a Agência do Patrimônio Cultural Catalão, em conjunto com o Museu d'Arqueologia de Catalunya-Ullastret designou juntamente com um estúdio de design gráfico e modelação 3D uma equipa multidisciplinar com especialistas em diferentes áreas (arquitetos, antropólogos, hidrogeólogos, construtores navais, etc.) com a intenção de definir com a máxima precisão e detalhe os múltiplos aspetos que devem ser tomados em conta para que a reconstrução alcançasse o máximo rigor científico.

A reconstrução foi baseada numa época específica na história da cidade, cerca de 250 a.C., devido este ser o período mais conhecido em termos científicos. O primeiro passo na reconstrução foi baseado numa compilação exaustiva de toda a informação arqueológica disponível. Em termos técnicos, o modelo foi baseado na base de dados Cartográfica 3D do Instituto Cartográfico e Geológico da Catalunha. Com estes dados foi possível gerar a malha geométrica para representar o relevo, introduzindo neste caso o antigo lago de Ullastret, que foi drenado no século XIX. Esboços tridimensionais foram feitos através das plataformas em softwares de modelação 3D, utilizados para a arquitetura e a paisagem. Como resultado a reconstrução virtual permite que os espectadores façam uma viagem pela cidade ibérica, passando pelas suas ruas e casas vazias. A história é narrada por um antigo habitante, um membro da elite, que lembra momentos dramáticos na história da cidade.

Figura 2.8: Reconstrução Virtual da antiga cidade de Ullastret [29]

Reconstrução Histórica da Cidade de Cagliari [30]

O projeto consiste na reconstrução histórica em 3D da cidade de Cagliari com as suas muralhas nos séculos XV, XVI e XVIII. Para a reconstrução histórica da cidade de Cagliari, primeiro foi considerado a estrutura das muralhas ainda visíveis nos dias atuais. Embora incorporados em edifícios dos séculos recentes ou parcialmente desmantelados, tais fortificações mostram a sua impressionante presença em vários pontos da cidade. Os dados foram obtidos a partir da cartografia histórica e de vários estudos já publicados.

As hipóteses finais de reconstrução mostram a cidade nos séculos XV, XVI e XVIII, respetivamente. A transformação sofrida pelas muralhas e torres medievais no século XVIII, como consequência da ampla difusão de armas de fogo, onde é claramente apresentada na simulação, que também mostra em detalhes como a cidade mudou ao longo dos séculos.

Carthago Nova [31]

Fundada pelos cartagineses como principal enclave na Península Ibérica e depois da sua conquista por Publius Cornelius Scipio "O Africano" no ano 209 a.C., Carthago Nova veio a se tornar uma das cidades mais importantes do Império Romano.

Do grande esplendor de Carthago Nova, os inúmeros restos arqueológicos que emergem na atual Cartagena testemunham, um passado de grande importância histórica e cultural.

O projeto Carthago Nova *"El esplendor de una era"* recriam virtualmente alguns dos cenários mais importantes daquela época. Os quais podemos destacar:

- **As termas de Carthago Nova**: Os banhos termais foram construídos no século I d.C. Não só representaram um lugar de banho e limpeza, como também eram um foro de reunião e diversão onde era possível estar com os amigos ou fechar acordos políticos e comerciais.
- **A casa Salvius:** datada do século I d.C., é a típica casa italiana das famílias romanas com grande poder aquisitivo, um símbolo de seu *status* e riqueza.

O projeto contempla ainda outros cenários recriados virtualmente da cidade naquela época, como por exemplo o porto, o anfiteatro e o fórum.

O Fórum Flaviano de Conimbriga [32]

O conjunto arquitetónico do Fórum Flaviano de Conimbriga foi reconstruído virtualmente utilizando a linguagem VRML. O projeto de reconstrução do Fórum teve por objetivo propagar o valor cultural do monumento, além de estabelecer uma nova forma de apreciação de sítios arqueológicos mediante acesso remoto, onde é possível a qualquer visitante "deslocar-se" no ambiente e, assim, aperceber-se melhor das suas características e das diversas perspetivas que era possível desfrutar sobre o Fórum real.

Museu Virtual Islâmico [33]

A cultura muçulmana também está ao alcance do público com o museu virtual sem fronteiras localizado num centro cultural no bairro islâmico de Al-Azhar no Egito. O projeto *Discover Islamic Art*[16] visa reproduzir antiguidades islâmicas guardadas em outros museus e fornecer explicação sobre essas peças a estudantes de arte, além de relacionar os textos históricos com as peças arqueológicas às quais correspondam.

Castelo de Pietrabuona [34]

Atualmente existem inúmeras pesquisas sobre a temática de reconstrução de espaços urbanos desaparecidos. Com isso, inúmeras aplicações têm sido desenvolvidas. Porém o foco, está voltado para um rápido desenvolvimento de ambientes urbanos inespecíficos, em vez de uma representação fidedigna da realidade de forma meticulosa e que seja percetível.

Neste cenário, o software 3D do SIUR (Sistema Informativo Urbano tridimensional) é baseado numa estrutura de gestão que liga um modelo interativo, foto realista e bastante fiável de uma cidade com uma base de dados qualitativa do histórico arqueológico. Essa plataforma utiliza o Unity 3D para gestão de modelos geométricos onde é possível fazer para compartilhamento de dados online. A arquitetura cliente/servidor possibilita o armazenamento de dados capturado numa base de dados, e que são disponibilizados numa interface web.

A Tabela 1 refere-se a uma comparação de projetos relacionados da aplicação da RV na HC.

Tabela 1: Relação de trabalhos inerentes a HC com a RV sem Humanos Virtuais.

Autores	**Ref.**	**Cenário de Reconstrução**	**Interação com Ambiente Virtual**	**Plataforma/Software utilizado**
Bernardes, P. (2002)	[27]	Cidade de Bracara Augusta (Portugal)	Sim	VRML 2.0
Sanchotene, I. S. (2007)	[26]	Coliseu de Roma (Itália)	Sim	MultiGen Creator[17]e Polytrans[18]
Codina *et al.* (2017)	[28]	Cidade de Ullastret (Espanha)	Sim	Cinema4D e 3D Studio Max

[16] Discover Islamic Art: http://islamicart.museumwnf.org/

[17] MultiGen Creator: https://www.modelbenders.com/ein5255/MG25_Tutor.pdf

[18] Polytrans: https://www.okino.com/default.htm

Município de Cagliari / SJM Tech (2008)	[30]	Cidade de Cagliari (Itália)	Sim	Unity 3D
Region de Murcia Digital (2011)	[31]	Termas de Carthago Nova (Espanha)	Sim	VRML
Region de Murcia Digital (2011)	[31]	Casa Salvius (Espanha)	Sim	VRML
Gonçalves *et al.* (2003)	[32]	Fórum Flaviano de Conimbriga (Portugal)	Sim	VRML
Grupo Estadão (2009)	[33]	Reconstrução de antiguidades (Egito)	Sim	Plataforma Web/html5 e Javascript com Base de dados
Troiano *et al.* (2014)	[34]	Castelo de Pietrabuona (Itália)	Sim	Unity 3D e software com arquitetura cliente/servidor

2.5.2 Simulações Históricas com humanos virtuais

Os Humanos Virtuais estão sendo utilizados numa variedade de aplicações relacionados com a herança cultural, tornando-as mais realistas, atraentes e interessantes.

Nesta secção apresentamos um conjunto de trabalhos de simulações históricas de reconstruções com humanos virtuais, dotados de animações pré-definidas e de simulações históricas com personagens (HV) autónomos dotados com algum tipo de Inteligência Artificial (IA). Existe uma grande variedade de implementações para os humanos virtuais em aplicações no que diz respeito ao património cultural, mais precisamente na reconstrução de ambientes históricos onde cada HV tem a sua aparência, diferentes capacidades de interação, várias tecnologias por trás da sua implementação. No entanto, é importante identificar as dependências existentes entre os objetivos do projeto e os principais recursos emergentes (o tipo de HV, interação com utilizador e a tecnologia de implementação), isso servirá para elaborar algumas diretrizes gerais e boas práticas sobre a utilização de humanos virtuais em projetos relacionados com a Herança Cultural.

Simulações históricas incluindo humanos virtuais com animações pré-definidas

A presente secção enquadra trabalhos de simulações históricas que usam a RV como plataforma e humanos virtuais dotados de animações que foram previamente definidas de acordo com o objetivo do projeto.

A igreja de Hagia Sophia [35]

A restituição virtual da igreja de Hagia Sophia (localizada em Istambul, Turquia) é um grande exemplo de utilização de HV na Herança Cultural Virtual pois, além da reconstrução virtual de edifícios e locais históricos, a experiência inclui HV representando pessoas de uma

certa era, a fim de aumentar o realismo da simulação e tornar a experiência do utilizador mais autêntica. Neste exemplo, ao representar os HV, os autores também levaram em consideração as particularidades históricas sobre o comportamento e a aparência dos personagens.

A catedral de St. Andrews [36]

O ambiente virtual denominado Open Virtual World desenvolvido pela University of St. Andrews, é uma plataforma aberta baseada em *OpenSim*, que é um conjunto de servidores distribuídos.

O projeto da Catedral de St. Andrews resgata o passado da catedral do séc. XIV e disponibiliza-o numa plataforma virtual. Ela permite aos visitantes participar em cenas históricas enquanto ouvem as histórias de personagens históricos que encontram, como o Bispo William de Lamberton, o Rei Robert I, e até mesmo os visitantes peregrinos. Cada um desses HVs tem uma história para contar e um cronograma de eventos. Em geral, o programa permite que os Humanos Virtuais possam interagir com os utilizadores, essa interação e feita através de um *script* que reage de acordo com os eventos, baseados em palavras-chave com os utilizadores, a resposta é feita por meio de texto ou fala.

Rome Reborn [37] [38]

O projeto *Rome Reborn*[19] é um dos maiores projetos de digitalização do mundo e funciona há mais de 15 anos. Os principais objetivos do projeto são produzir uma versão em alta resolução de Roma em 320 dC, um modelo de menor resolução para criar um aplicativo *mashup* com o *Google Earth*[20] e, finalmente, o modo colaborativo do modelo para uso com aplicação do mundo virtual e voltado principalmente para a educação [37]. Em particular o *Rome Reborn*, visa investigar a adequação do uso desta tecnologia para apoiar a exploração arqueológica de aspetos societários historicamente precisos da vida de Roma, com ênfase em expressões políticas, religiosas e artísticas.

Para alcançar esses objetivos, o projeto integra quatro tecnologias de ponta do mundo virtual com o modelo *Rome Reborn*, o modelo tridimensional mais detalhado da Roma Antiga disponível. Entre elas está o uso do mecanismo de vida artificial *Instinct* [38], que permite a animação coerente de multidões e, portanto, cria uma simulação da população da cidade de Roma com personagens virtuais com uma variedade de comportamentos. Esses personagens virtuais com diferentes comportamentos podem ensinar o utilizador sobre diferentes aspetos da vida em Roma (condições de vida, política, militar) [38].

O projecto *Rome Reborn* utiliza também o software CityEngine que possui uma versão free trial de trinta dias e a sua principal característica é a extensibilidade de novas funcionalidades por regras gramaticais. Essa regras são estabelecidas e pré-definidas com base em consulta de especialistas (arqueológicos) para recolha de dados precisos e credíveis, o ambiente virtual é caracterizado pela disponibilidade pela Web, o que permite ao utilizador aceder e interagir com a reconstrução da cidade antiga de Roma.

Okapi Island [39] [40]

Okapi Island (SLurl) é um ambiente virtual imersivo projetado para proporcionar aos visitantes uma experiência em 3D de Çatalhöyük no centro da Turquia, o local de uma

[19] Rome Reborn: https://www.romereborn.org/

[20] *Google Earth*: https://www.google.com/earth/

comunidade agrícola neolítica que floresceu de 9.400 até 7.700 anos atrás. Faz parte do projeto Remixing Çatalhöyük da Universidade da Califórnia, em Berkeley, e é apresentado pelos avatares Ruth Galileo e Zed Marseilles. O projeto é apoiado pelo Conhecimento Aberto e o Interesse Público (OKAPI), uma organização que reúne professores, alunos e funcionários para promover o conhecimento aberto e a cultura livre em Berkeley juntamente com o Departamento de Antropologia em Berkeley.

A Ilha Okapi é inspirada no East Mound em Çatalhöyük, e representa o local como existe hoje e como pode ter sido no passado. Os visitantes são convidados a participar de vários espaços e recursos, incluindo um passeio de vídeo guiado, o Museu Virtual de Okapi, reconstruções em 3D de Çatalhöyük, uma área de escavação virtual e a parede do graffiti.

Através do projeto Okapi Island os alunos da universidade de Berkeley durante o outono de 2009, criaram o "machinima" que foi filmado na *Okapi Island*, uma recriação da vila Neolítica de 9000 anos de Çatalhöyük, Turquia; onde simularam animações da caça de um auroque, um ancestral do gado doméstico, assim como a morte acidental resultante de um dos caçadores, e a sua subsequente cerimônia funerária.

Virtual Hadrian's Villa Walkthrough [41]

O Laboratório do Instituto de Artes Digital da universidade *Ball State* no estado de Indiana (IDIA Lab[21]) projetou uma simulação virtual da vila do imperador romano Adriano, que é Património Mundial da UNESCO localizado fora de Roma, em Tivoli, Itália. Este projeto foi produzido em colaboração com o Laboratório do Património Mundial Virtual (VWHL) da Universidade de Indiana (IU). Esta recriação em grande escala interpreta todo o complexo da vila de acordo com os principais estudiosos desta vila histórica. O projeto foi criado na plataforma de jogo da *Unity3D* como um ambiente de aprendizagem on-line virtual multiutilizador em tempo real que permite que estudantes e visitantes mergulhem em todos os aspetos da vila simulada.

O projeto não apenas recria os prédios das vilas, mas também inclui um completo sistema de avatares estilo romano, com personagens dotados de inteligência artificial, com mobílias da época, vegetação apropriada, sistema atmosférico dinâmico e uma interface sofisticada. A interface permite que os utilizadores mudem a posição do sol para qualquer data em 130 dC usando a base de dados Horizontes da *JPL NASA*[22], com isso é possível testar as teses de astro-alinhamentos de características arquitetónicas durante solstícios e equinócios. Antes de mergulharem no sistema, os utilizadores aprendem sobre o ambiente virtual onde são informadas sobre a cultura e a história da vila. O sistema de avatares permite também que os visitantes entrem no mundo selecionando classe e género, já sabendo dos costumes e comportamento das classes sociais romanas: soldado, escravo ou político.

Simulações históricas com personagens autónomos

A credibilidade das simulações históricas em 3D com personagens autónomos depende, em grande parte, de sua capacidade de mostrar comportamentos plausíveis. Nesse sentido, as ações dos indivíduos dentro de grupos de seres humanos não são homogéneos nem periódicos e, consequentemente, a heterogeneidade e a espontaneidade dos comportamentos realizados pelos habitantes gerados por programação estão entre os fatores-chave para a qualidade e realismo das simulações, nesse sentido a presente secção enquadra trabalhos de simulações históricas que usam

[21] http://idialab.org/

[22] JPL NASA: https://www.jpl.nasa.gov/

a realidade virtual como plataforma e humanos virtuais animados com algum tipo de Inteligência Artificial, sendo estes os trabalhos que mais se assemelham ao trabalho realizado nesta tese.

Ancient Pompeii [42] [43]

Pompeia foi uma cidade romana destruída e completamente enterrada na primeira erupção registada do vulcão Monte Vesúvio em 79 dC. Para este projeto, um modelo da antiga Pompeia foi construído baseado em dados arqueológicos, para simular a vida antiga de Pompeia, para isso foram utilizados métodos procedurais, e posteriormente preenchido com avatares para simular a vida em Pompeia em tempo real.

O principal objetivo deste projeto era simular uma multidão de romanos virtuais exibindo comportamentos realistas dentro da cidade de Pompeia. A simulação permite também que os humanos virtuais possam navegar livremente em vários edifícios no modelo da cidade assim como interagir com o ambiente envolvente.

George Town [44]

Muitos esforços foram realizados para preservar a história e a cultura de Penang e também de outras regiões da Malásia, desde que George Town foi eleita como uma cidade patrimônio da UNESCO. Este projeto apresenta um método para simular a vida em um porto comercial local em 1800, onde várias populações com regras sociais muito diferentes interagem entre si. Essas populações incluíam indianos, vendedores malaios, colonos britânicos e comerciantes chineses. O desafio principal do projeto em si, foi modelar esses diferentes grupos étnicos como agentes autónomos e capturar as mudanças de comportamento devido às interações étnicas, assim como na chegada de barcos no píer. Os HV de cada população são equipados com um conjunto específico de métodos de direção que são selecionados e parametrizados de acordo com padrões comportamentais predefinidos (gráficos de estados).

O projeto em si resume-se, numa aplicação móvel que possibilita um passeio virtual interativo da cidade com duas diferentes abordagens: presente e passado. A ênfase aqui é colocada na simulação da dinâmica de multidões que é gerado de duas maneiras diferentes: no presente, imagens de 360 ° da cidade atual são povoadas por imagens 2D de HV que se comportam de acordo com informações recuperadas do vídeo (por exemplo: o reconhecimento de obstáculos), enquanto que na abordagem do passado, reflete-se a reconstrução da cidade antiga, juntamente com a multidão de seus habitantes, é totalmente virtual e, portanto, ocorre num ambiente digital 3D.

A cidade de Uruk [45]

A maioria dos aplicativos existentes relacionados com a herança cultural virtual está focada na reconstrução 3D detalhada de lugares historicamente significativos assim como artefactos antigos. Recriar o modo de vida dos povos antigos é apenas considerado por alguns investigadores, que empregam a simulação de multidões com este intuito. Neste contexto, surgiu este projeto de pesquisa do protótipo da cidade de Uruk, realizado em conjunto entre a Universidade de Western Sydney e a Federação de Cientistas Americanos, cujo o principal objetivo era recriar a antiga cidade de Uruk no período cerca de 3000 a.C. no mundo virtual 3D, onde foi utilizado a combinação da Realidade Virtual e a Inteligência Artificial para simular a vida quotidiana na primeira cidade da humanidade. O projeto foi concebido para fins pedagógicos, onde permitia que os estudantes de história experimentassem como era e como os cidadãos se comportaram no passado proporcionando-lhes a possibilidade de mergulhar numa réplica exata

da vida quotidiana dos antigos cidadãos de Uruk e obter uma rápida compreensão do avanço do desenvolvimento tecnológico e cultural dos antigos sumérios. Em última análise, os alunos podem se tornar parte da sociedade virtual, podendo interagir com agentes e outros seres humanos para resolver as tarefas atribuídas.

A reconstrução 3D da cidade foi produzida dentro do mundo virtual de *Second Life*[23] baseado nos resultados de escavações arqueológicas e fontes escritas disponíveis. Tanto a modelação da cidade quanto a programação dos humanos virtuais foram realizados sob a supervisão de especialistas no assunto.

Foram selecionados a vida quotidiana dos pescadores da antiga Uruk para ilustrar como as instituições podem permitir uma experiência imersiva na vida das sociedades antigas. Foram criados quatro agentes que representam membros de duas famílias de pescadores onde cada família consiste em marido e mulher. Todos os agentes têm uma única aparência historicamente autêntica e estão vestidos adequadamente com roupas da época.

Ágora Virtual [46]

A Ágora era um espaço aberto que existiu em várias cidades da Antiguidade, tendo atingido a sua maior expressão em Atenas. Ocorriam aí os mais variados encontros entre cidadãos que desenvolviam diversas atividades, nomeadamente política, filosofia, comércio, artes. Baseando-se nessa premissa foi criado o projeto da Ágora Virtual, uma aplicação da Herança Cultural Virtual onde demonstra as atividades da vida diária e rituais na Grécia antiga.

O objetivo principal do projeto é apresentar a vida na Grécia antiga e educar os utilizadores sobre as atividades comuns diárias e importantes rituais religiosos. A configuração da Ágora virtual é inspirada em áreas reais da Ágora e inclui edifícios tipicamente da época contidos num fórum antigo clássico. O lugar principal da Ágora é o Templo de Hefesto em Atenas, junto com um altar sacrificial, modelado de acordo com medições reais nas ruínas. A implementação do projeto foi baseada numa plataforma genérica multiagente, que se concentra no comportamento dinâmico, duradouro e credível dos personagens. A plataforma permite ainda que os designers facilmente possam estender e ajustar cenários existentes, ou até mesmo criar novos e estender funcionalidades.

A aplicação segue um modelo motivacional multicamadas para agentes que incluem necessidades biológicas, bem como psicossociais. Todo o agente possui um conjunto de atributos básicos que se relacionam com suas características biológicas e físicas, bem como com sua personalidade. Além disso, os agentes são dotados de um conjunto de comportamentos que satisfazem metas particulares que consistem em uma sequência de ações para alcançar objetivos. Além deste conjunto de ações genéricas, cada agente possui um conjunto extra de ações, com base em sua atribuição de um papel ou profissão. Os papéis e os respetivos comportamentos foram concebidos e selecionados com base nos recursos disponíveis sobre a vida na antiga Ágora da Atenas clássica.

Projeto Mediaevo [47]

O projeto MediaEvo é um jogo didático sobre a história da Idade Média e tem como público alvo professores e alunos. O projeto visa desenvolver uma plataforma multicanal e multissensorial na área da Herança Cultural Virtual e testar novas tecnologias de processamento de dados para a realização de um jogo didático digital orientado para o conhecimento da história e sociedade medievais.

[23]Second Life: https://secondlife.com/

Otranto é uma cidade está localizada no sul da Itália, na península de Salento e desempenhou um papel importante na Idade Média do ponto de vista histórico e cultural. Devido a sua posição geográfica no extremo leste da Itália, Otranto era como uma ponte entre o leste e o oeste do Mar Mediterrâneo.

O projeto que reconstrói o ambiente desta cidade possui características de jogos de estratégia, em que as capacidades de decisão de um utilizador têm um grande impacto no resultado, que no caso seria a conquista de um alvo de aprendizado. A ideia é fornecer uma competição entre os jogadores, durante a aprendizagem. O sistema, com base numa meta de aprendizagem bem definida, proporá continuamente novos caminhos para permitir a obtenção de diferentes resultados de aprendizagem.

Para a modelação do terreno de jogo, foi utilizado a ferramenta ESRI ArcGIS e o motor de jogo *Torque Game Engine* do *GarageGames*[24], já para criar os conteúdos arquitetónicos 3D, foi utilizado o editor *Torque Constructor* do motor Torque 3D. A Inteligência Artificial foi implementada através de um algoritmo, com a capacidade de estabelecer conexões textuais ou vocais estáveis entre os diferentes jogadores virtuais colocados na missão do jogo.

A Tabela 2 refere-se a uma comparação de projetos relacionados da aplicação de HVs na HC, onde os critérios escolhidos ajudam a fornecer uma visão comparativa dos projetos pesquisados sob vários pontos de vista: objetivo e tipo de aplicação, usabilidade, tecnologia por trás da implementação e participação do utilizador. Foi adicionado uma perspetiva sobre como tais plataformas evoluíram ao longo dos anos, a fim de ter uma melhor imagem das principais tendências.

Tabela 2: Avaliação comparativa de projetos de Herança Cultural que incluem Humanos Virtuais.

Projeto	**Ref/ Ano Publicação.**	**Tipo**	**Humano Virtual**	**Interação com utilizador**	**Plataforma/ Software utilizado**
A igreja de Hagia Sophia.	[35] 2002	Ambiente Virtual 3D	Multidão (pessoas)	Não é possível	Modelos 3D criados utilizando o 3D Studio Max 4.2
A catedral de St. Andrews.	[36] 2013	Ambiente Virtual 3D	Único (avatar do utilizador mostrado na perspetiva de primeira ou terceira pessoa) + figurantes que povoam a igreja.	O Utilizador pode interagir com os figurantes que povoam o ambiente em conversas textuais e auditivas.	Open Virtual Worlds; OpenSim; Oculus Rift

[24]GarageGames: http://www.garagegames.com/products/torque-3d

Rome Reborn	[37] [38] 2011	Ambiente Virtual 3D	Multidão (cidadãos romanos)	Não é possível.	CityEngine
Okapi Island	[39] [40] 2009	Ambiente Virtual 3D	Multidão (pessoas)	Não é possível	Plataforma SecondLife
Virtual Hadrian´s Villa Walkthrough	[41] 2012	Ambiente Virtual 3D	Multidão (cidadãos romanos)	O Utilizador interage em vários cenários, assim como pode interagir com avatares	Modelos 3D criados através do 3D Studio Max e montados no Motor de jogos Unity 3D.
Ancient Pompeii	[42] [43] 2007	Ambiente Virtual 3D	Multidão autónoma (cidadãos romanos)	Não é possível	Modelos 3D criados no CityEngine com gráfico de navegação de dados semânticos
Georgetown	[44] 2016	Ambiente Virtual 3D	Multidão autónoma (habitantes da cidade)	Os humanos virtuais interagem em cenários avançados, mas nenhuma interação com o utilizador é possível.	Unity 3D
A cidade de Uruk	[45] 2009	Jogo Sério Ambiente Virtual 3D	Multidão autónoma (habitantes da cidade)	Q & A interação baseada em chat.	o Design 3D da cidade foi feito utilizando Unity 3D.
Ágora Virtual	[46] 2016	Ambiente virtual 3D imersivo	(Avatar do utilizador) + (pessoas andando pela praça)	O avatar do utilizador pode interagir com outros avatares, através de conversas, histórias, missões e outras atividades.	Open Virtual Worlds; OpenSim

Projeto Mediaevo	[47] 2011	Jogo Sério Ambiente Virtual 3D	Multidão (habitantes da cidade)	O Utilizador pode jogar como o avatar	Nintendo Wiimote and Balance Board

2.6 Conclusão

Neste capítulo foram apresentados um conjunto de conceitos base e trabalhos relacionados considerados essenciais no contexto do presente projeto. Foram descritos os conceitos de Realidade Virtual e Humanos Virtuais assim como as suas áreas de aplicação.

Através da leitura dos artigos pesquisados percebemos que para resgatar um sítio antigo totalmente desaparecido é preciso investigar o seu passado histórico e identificar registos para elaboração de modelos tridimensionais credíveis. Neste contexto, são necessários, um conjunto de profissionais nomeadamente arqueólogos e historiadores que são extremamente importantes para identificar realidades no passado, e assim possibilitar a transição desta informação para uma plataforma virtual onde será disponibilizada para gerações presentes e futuras. Baseando-se nessa premissa, foram apresentadas também as definições de Herança Cultural e Herança Virtual.

Apresentaram-se também alguns trabalhos de simulações históricas e a sua relação com a Realidade Virtual, sem a utilização de humanos virtuais e exemplos de simulações históricas com humanos virtuais, animados com algum tipo de Inteligência Artificial que se assemelham a ferramenta que foi desenvolvida no âmbito deste projeto.

No capítulo seguinte, é descrita a abordagem tomada para o desenvolvimento do projeto.

Capítulo 3

Análise de ferramentas e caso de estudo

Neste capítulo descreve-se o trabalho realizado referentes ao estudo de ferramentas de software considerados adequado ao contexto do projeto que foi desenvolvido, referencia-se também a interoperabilidade de dados entre as ferramentas escolhidas, assim como os recursos envolvidos no desenvolvimento do processo de criação do cenário de Mértola Virtual, as versões das ferramentas utilizadas, e os requisitos mínimos para a execução da mesma. Introduz-se também caso de estudo para o cenário de Mértola Virtual, assim como o *pipeline* que foi aplicado para a construção do referido ambiente urbano virtual realizado pelo projeto do Alexandre Carvalho [3].

3.1 Relação das ferramentas de software

A presente secção possui como objetivo classificar as ferramentas de software quanto à sua utilização no âmbito da herança cultural digital. As escolhas das ferramentas foram baseadas em vários aspetos, nas quais podemos destacar: a gratuidade, a compatibilidade do fluxo de dados entre elas, fontes disponíveis para consulta (manuais), todas possuem pelo menos um dos aspetos anteriormente mencionados, além de outras características importantes necessárias para a criação do cenário Virtual entre elas: a modelação tridimensional e renderização, modelações de personagens, animação, e conversão de formatos.

Estudaram-se ferramentas de domínio publico (Blender), ou que têm um subconjunto de funcionalidades gratuitas (Daz Studio, 3DS Max, Google SketchUp, Mixamo, Adobe CC Photoshop) ou para as quais existe uma licença académica (Unity 3D). Descrevem-se em seguida cada uma dessas ferramentas:

- *Blender*[25] : é uma aplicação de modelação 3D, que é utilizada para modelar e editar os elementos necessários para aplicação, permitindo a importação e exportação de diversos formatos. Entre os quais cabe destacar formato OBJ e FBX. A versão estudada é a 2.76.
- *Daz Studio*[26]*:* é um software completo para criação de imagens 3D. Nele podemos criar e renderizar modelos para planos de fundo, objetos e personagens 3D. O programa é capaz de importar e exportar, animações, objetos e personagens 3D de outros softwares.
- *3DS Max*[27]: ferramenta de modelação tridimensional, onde é possível, gerar rapidamente personagens realistas, animações e efeitos de computação gráfica. O programa se caracteriza pela sua versatilidade na aplicação de texturas, assim como na renderização de imagens. O formato FBX permite a interoperabilidade de dados com outras ferramentas, como por exemplo, o Unity 3D.

[25] Blender: http://www.blender.org/

[26] Daz Studio: http://www.daz3d.com

[27] 3DS Max: https://www.autodesk.com/products/3ds-max/features

- *SketchUp*[28]: ferramenta de modelação 3D que privilegia a facilidade de utilização em detrimento da complexidade dos objetos3D, tornando-se, como o nome sugere, bastante útil na modelação de protótipos e maquetes.
- *Adobe CC Photoshop*[29]: ferramenta utilizada para editar texturas/imagens de elementos 3D, assim como pode ser utilizada para a conversão de formatos, garantindo assim a interoperabilidade dos dados 3D.
- *Mixamo*[30]*:* aplicação para a criação de humanos virtuais, com uma variedade de atributos e caraterísticas que podem ser alterados, é possível também manipular e animar os personagens. A criação e modelação dos personagens é feita no *Mixamo Fuse*[31].
- *Unity3D*[32]: motor de jogo, utilizado principalmente para o desenvolvimento de videojogos, nela é possível definir o comportamento dos objetos da cena através de *script* C#, ambiente de programação nativa integrada na plataforma Unity 3D, entre outras particularidades, torna-se possível criar efeitos gráficos, controlo do comportamento físico de objeto e de personagens através do motor de física. Admite a importação de dados tridimensionais desenvolvidos em diversas ferramentas. Neste caso o Unity3D vai ser utilizado para o desenvolvimento da aplicação, podendo assim incorporar os modelos 3D importados de diversas aplicações.

3.1.1 A interoperabilidade de dados entre ferramentas

A interoperabilidade é uma característica de um produto ou sistema, cujas interfaces são completamente compreendidas, para trabalhar com outros produtos ou sistemas, presentes ou futuros, na implementação ou no acesso, sem quaisquer restrições. Com relação ao *software*, o termo interoperabilidade é usado para descrever a capacidade de diferentes programas de trocar dados através de um conjunto comum de formatos de troca, ler e gravar os mesmos formatos de ficheiros e utilizar os mesmos protocolos.

Segundo Shen et al. [48], a interoperabilidade traduz a capacidade de certos dados gerados por uma parte específica serem corretamente interpretados por outras partes. Segundo os autores, esse é o primeiro passo para qualquer integração de sistemas e colaboração.

Em outras palavras a interoperabilidade procura proporcionar compatibilidade entre as diferentes aplicações com a transferência de dados mantendo a sua integridade. Tendo em vista, a diversidade de aplicações necessárias que são utilizadas na reconstrução de herança cultural digital, a garantia da interoperabilidade de dados entre elas, dar-se-á através dos formatos de exportação e importação. Uma premissa na escolha das ferramentas que foram adotadas tanto no estudo como na utilização delas, foi a compatibilidade dos formatos de exportação/ importação.

Na Tabela 3 apresentam-se os principais formatos de importação e exportação entre as ferramentas de *software* analisadas.

---

28 SketchUp: https://www.sketchup.com/

29 Adobe CC Photoshop: https://www.adobe.com/pt/products/photoshop.html

30 Mixamo: http://www.mixamo.com

31 Mixamo Fuse: https://www.adobe.com/products/fuse.html

32 Unity3D: https://unity3d.com/

Tabela 3: Troca de dados entre ferramentas.

Software	**Formato de importação**	**Formato de Exportação**
Blender	DAE, 3DS, FBX, BVH, PLY, OBJ, X3D, STL, SVG	DAE, 3DS, FBX, PLY, OBJ, X3D, STL
Daz Studio	BHV, DAE, FBX, OBJ, PZ3, PZ2, CR2, IT2, CM, PZZ, P2Z, CRZ, ITZ, CMZ, PPZ, FCZ, HDZ, OBZ	BHV, DAE, CR2, FBX, MI, OBJ, U3D, MDD
3DS Max	FBX, DWG e DXF, OBJ	DXF, FBX, MI, OBJ
Adobe CC Photoshop	3D STUDIO DAE, JPS, KMZ, MPO, U3D, WAVEFRONT OBJ, JPEG, BMP, GIF, PNG	DAE, FLASH 3D, JPS, KMZ, MPO, WAVEFRONT OBJ, RAW, JPEG, BMP, GIF, PNG
SketchUp	DWG, DXF, JPG, PNG, TIF, BMP, PSD, TGA, PDF, EPS, STL, DAE, 3DS, DEM, DDF	DWG, DXF, JPG, PNG, TIF, BMP, PSD, TGA, PDF, EPS, STL, DAE, 3DS, FBX,KMZ, OBJ, WRL, XSI, IFC
Mixamo	FBX, OBJ, ZIP	FBX, DAE
Unity 3D	FBX, DAE, 3DS, DXF, OBJ, RAW	FBX, RAW

De acordo com a Tabela 3, existem algumas limitações das ferramentas no que se refere a importação e exportação de dados, onde em alguns casos é possível importar um determinado ficheiro com um formato específico e não podendo posteriormente exportá-lo com o mesmo formato. Baseando-se nesse argumento, para o projeto optou-se por adotar ferramentas que fossem ao máximo compatíveis entre si, ou seja, ferramentas que adotassem o mesmo tipo de formatos tanto para importação quanto para exportação, garantindo assim o fluxo da informação. Como o Unity3D vai ser utilizado para o desenvolvimento da aplicação, os formatos escolhidos por serem comum em todas ferramentas é o FBX e/ou OBJ.

Na Figura 3.1 apresentam-se os formatos de ficheiros de dados que podem ser utilizados para fazer a transferência de dados 3D entre as ferramentas de software analisadas. Estas ferramentas podem associar-se às diferentes etapas de reconstrução de herança cultural digital, desde a modelação tridimensional até renderização.

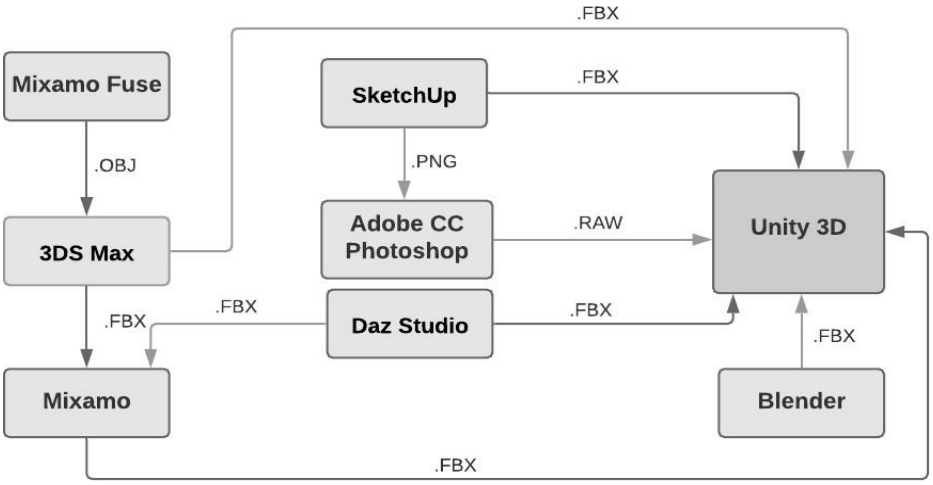


Figura 3.1: Fluxo de dados compatíveis entre as ferramentas

3.1.2 Requisitos Técnicos

Nesta secção são apresentados os recursos envolvidos no desenvolvimento do processo de criação do cenário de Mértola Virtual, as versões das ferramentas utilizadas, assim como os requisitos mínimos para a execução da mesma.

A Tabela 4 resume as principais características das ferramentas de software que foram escolhidas, levando em consideração as etapas necessárias para a criação do cenário Virtual (modelação tridimensional e renderização), a gratuidade das ferramentas (domínio), permissão de fluxos de dados entre si (interoperabilidade de dados 3D), fontes diversas de aprendizagem (manuais e tutoriais), modelações de personagens e animação, conversão de formatos.

Tabela 4: Ferramentas escolhidas.

Ferramentas	Modelação 3D	Domínio	Fluxo de dados 3D	Manuais e Tutoriais	Modelação de Personagens e animação	Conversão de formatos
Blender	Sim	Público. Versão gratuita	Sim	Sim[33]. Existem várias fontes de formação	Sim	Sim
Daz Studio	Sim	Versão gratuita.	Sim	Sim.[34]	Sim	Sim

[33]Manual do blender: https://docs.blender.org/manual/pt/2.79/index.html

[34] Vídeos tutoriais: https://www.daz3d.com/help/help-daz-3d-video-tutorials

3DS Max	Sim	Versão de teste gratuito.	Sim	Sim[35]. Existem várias fontes de formação	Sim	Sim
Adobe CC Photoshop	Sim	Versão de teste gratuito. Período limitado	Sim	Sim[36].	Não	Sim
SketchUp	Sim	Versão de teste gratuito. Período limitado	Sim	Sim[37].	Permite modelação de personagens e não permite animação.	Sim
Mixamo/ Mixamo Fuse	Sim	Versão de teste gratuito. Período limitado	Sim	Sim[38].	Sim	Sim
Unity 3D	Sim	Versão gratuita e paga.	Sim	Sim[39].	Sim	Sim

Tendo em conta a preocupação de obter uma solução de baixo custo como sendo um dos objetivos do projeto é essencial recorrer na medida do possível a ferramentas gratuitas ou com licenças académicas e que permitam fluxos de dados entre si, e estas ferramentas possuem as características necessárias que cumprem essa premissa, além de possuírem também as características essenciais paras as etapas do processo de criação da ferramenta do cenário de Mértola Virtual Além das ferramentas já mencionadas, foi necessário recorrer a plataforma Arteria3d[40] para obtenção de alguns animais com suas respetivas animações, assim como a *assetstore* do Unity3D[41].

A Arteria3d é a plataforma que fornece conteúdo 3D e músicas de qualidade para jogos de computador, educação e mercados comerciais, sendo um dos mercados principais da Arteria3d os

[35]Autodesk: http://help.autodesk.com/view/3DSMAX/2016/ENU/

[36] Tutoriais: https://helpx.adobe.com/photoshop/tutorials.html

[37] Vídeos tutoriais e ajuda: https://help.sketchup.com/en

[38]Mixamo help: https://helpx.adobe.com/creative-cloud/help/animate-characters-mixamo.html

[39]Unity Manual: https://docs.unity3d.com/Manual/index.html

[40]Arteria3d: https://arteria3d.myshopify.com/

[41] assetstore: https://assetstore.unity.com/

conteúdos medievais, é possível obter modelos gratuitos, totalmente compatíveis com várias ferramentas, porém na sua grande maioria é necessário comprar.

Ao realizar alguns testes viu-se a necessidade de manipular algumas texturas, assim como, redimensioná-las para isso, foi a utilizada a ferramenta Gimp[42], que é uma ferramenta gratuita multiplataforma de manipulação de fotos, ela é perfeitamente adequada para várias tarefas de manipulação de imagens, incluindo retocar fotos, composição e construção de imagens.

Versões das ferramentas adotadas:

- **Blender**: versão 2.76.
- **Daz Studio**: 4.8 Pro.
- **3DS Max:** versão 2016 académica.
- **Adobe CC Photoshop**: versão de 2016 (30 dias gratuitos).
- **SketchUp**: versão 2016 profissional (30 dias gratuitos).
- **Mixamo**: Versão 2016.
- **Mixamo Fuse**: Versão 1.0
- **Unity 3D**: Versão 5.6.1
- **Gimp**: versão 2.8.16.

Requisitos mínimos para a execução da aplicação Mértola Virtual:

- **Sistema operativo**: a aplicação está preparada para ser executada em diferentes ambientes e plataformas (e.g. Windows, OSX).
- **Processador (CPU):** 2 GHz dual-core CPU mínimo (Intel Pentium 4 / AMD Athlon 64 ou Superior).
- **Placa Gráfica (GPU):** ATI Radeon 9550 / NVidia GeForce FX 5500 ou superior.
- **Memória RAM**: 8GB ou mais (16 GB recomendável).
- **Espaço em disco**: 1 GB ou mais.

3.2 A Reconstrução de ambientes históricos desaparecidos, o cenário virtual da vila de Mértola

Portugal é um país com uma longa história e um extenso e rico património arquitetónico e cultural. A vila de Mértola, situada no distrito de Beja, na região do Alentejo, foi escolhida como o caso de estudo. O passado de Mértola e os seus vestígios têm, desde há muito, chamado a atenção de viajantes, arqueólogos e historiadores.

A reconstrução do cenário urbano virtual da vila de Mértola, foi baseada no período da ocupação Islâmica, entre os séculos VII e XII, tendo como base a consulta de desenhos e esboços do espaço histórico, artigos de arqueólogos e historiadores, assim como registos históricos [54].

A mando do rei D. Manuel I, Duarte de Armas [49], foi enviado para catalogar as fortalezas de Portugal no século XVI. Embora o desenho deste autor não se reporte à época relativa ao trabalho (século XI) é provável que as muralhas de Mértola tenham mantido as suas dimensões.

Construída sobre um imponente esporão, Mértola assumiu uma posição estratégica excecional. Ao analisarmos pormenorizadamente o desenho desta cidade feito por Duarte de Armas na Figura 3.2 rapidamente nos apercebemos das potencialidades da sua fortificação e até da forma como esta localização foi escolhida, situando-se entre o rio Guadiana e o Oeiras, a

[42] Gimp: https://www.gimp.org/

Figura 3.2 identifica também a Torre do Rio e uma das portas de entrada da Vila representando Mértola naquela época, é notável que dentro das muralhas havia zonas consideráveis sem construções (espaços vazios ou de hortas), nomeadamente as que apresentam um terreno mais íngreme.

Figura 3.2: Mértola fortificada vista de Sueste [49]

Em comparação ao registo de Duarte de Armas, a Figura 3.3 mostra uma vista área recente de Mértola.

Figura 3.3: Vista área de Mértola e do rio Guadiana [50]

Alcáçova de Mértola

Segundo o trabalho efetuado por Santiago Macias [51], a Alcáçova de Mértola caracterizou-se por um local onde reinava a organização urbana. Fruto de uma construção de raiz, embora reutilizando alguns materiais e estruturas do período romano, o Bairro da Alcáçova edificou-se e cresceu entre os séculos XI e XII, segundo o mesmo estudo e de e acordo com as escavações levadas a cabo pela equipa de Santiago Macias, o conjunto habitacional da Alcáçova de Mértola, rondava as três dezenas de casas. Constituída por vários compartimentos, as moradias eram padronizadas no estilo clássico mediterrânico. Em volta de um pátio interior, organizavam-se os vários compartimentos da casa; o telhado da casa estava inclinado para o pátio interior, a casa muçulmana encontrava no pátio o seu lugar central.

Na Figura 3.4 apresenta-se uma vista em 3 dimensões das habitações do Bairro da Alcáçova de Mértola.

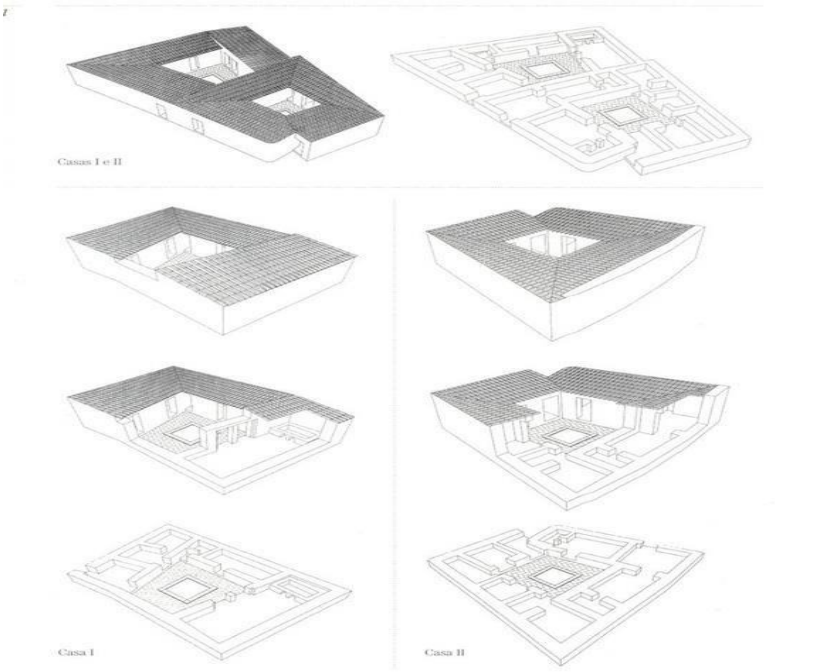


Figura 3.4: Vista em 3D das habitações do Bairro da Alcáçova de Mértola [50]

3.3 Sequência de etapas realizadas para criação e integração do cenário virtual de Mértola

Tendo por base principal informação fornecida por arqueólogos do Campo Arqueológico de Mértola, foi proposto e testado um *pipeline* que permitiu a construção do ambiente urbano virtual da vila de Mértola na época de ocupação Islâmica [3]. Este *pipeline* combina várias ferramentas gratuitas ou com licenças académicas, recorrendo sempre a interoperabilidade de dados 3D.

Para o processo de desenvolvimento do *pipeline* foram consideradas duas etapas: a modelação do terreno e a modelação dos edifícios. O objetivo final deste *pipeline* era que fosse

viabilizada a importação de todo o modelo para Unity3D, onde posteriormente serão incorporados personagens virtuais, tirando partido de algoritmos de inteligência artificial.

De acordo com a Figura 3.5, representa-se as etapas que foram necessárias para a geração do ambiente virtual.

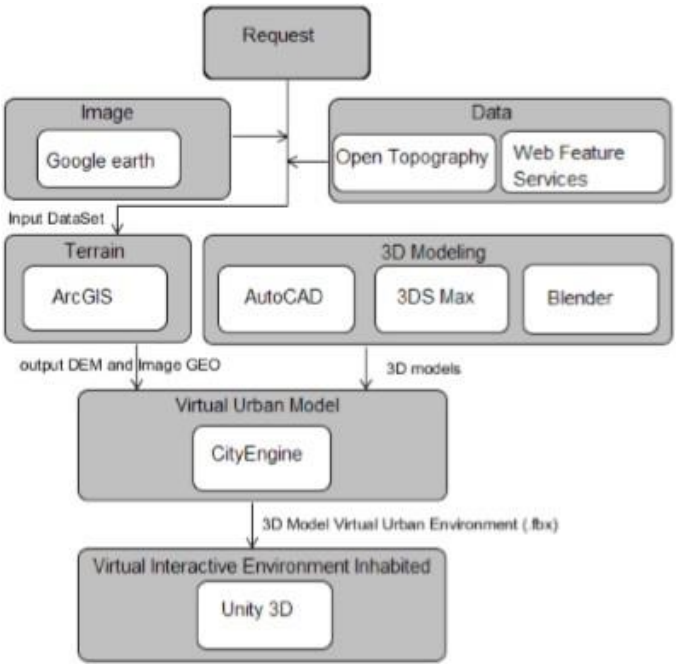


Figura 3.5: Pipeline para geração do ambiente urbano virtual [3]

Para a aquisição e processamento de informação geográfica e do relevo visando obter um modelo preciso do terreno foi utilizada a ferramenta ArcGIS.

As modelações das estruturas arquitetónicas foram realizadas pelas ferramentas AutoCAD, Blender e 3DS Max, já para a geração semi-automatizada de estruturas 3D através de ficheiros de regras, permitindo assim a possibilidade de inserção no referido cenário uma vasta diversidade de modelos tridimensionais, foi o utilizada a ferramenta CityEngine.

O modelo resultante desta combinação foi exportado para Unity 3D e a imagem final pode ser observada na Figura 3.6.

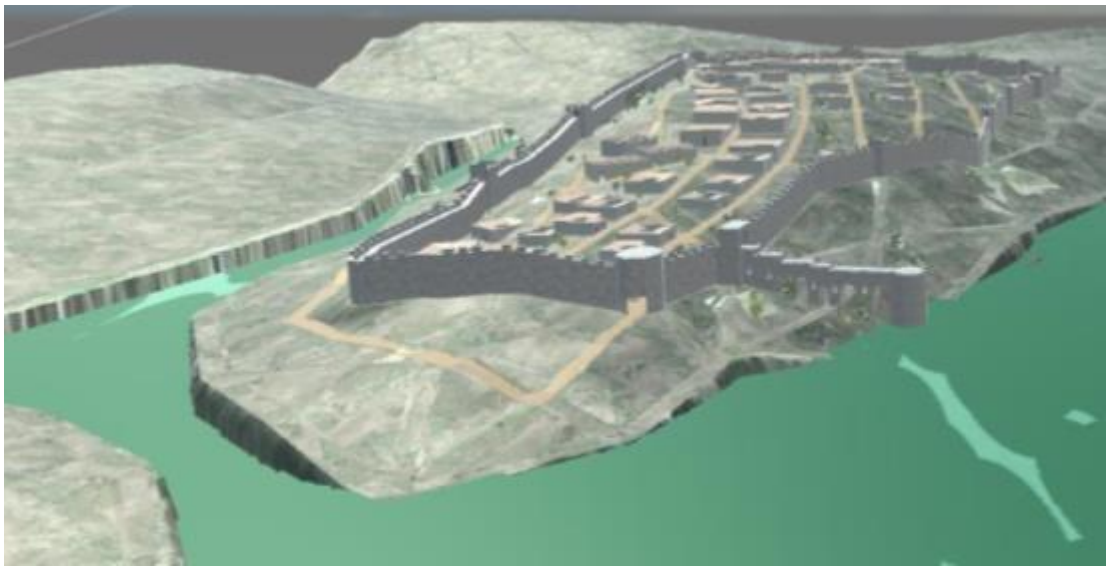


Figura 3.6: Modelo final exportado do CityEngine para Unity 3D [3]

3.4 Conclusão

Neste capítulo foi descrito um conjunto de ferramentas que foram consideradas adequadas ao contexto do projeto, referenciou-se a interoperabilidade de dados entre as ferramentas como uma importante característica na escolha das ferramentas. Indicaram-se as versões das ferramentas utilizadas no desenvolvimento do projeto assim como os requisitos mínimos para a execução do mesmo.

No próximo capítulo será descrito o desenvolvimento da ferramenta Mértola Virtual, bem como as etapas que a originaram.

Capítulo 4

Ferramenta Mértola Virtual

O contributo principal deste trabalho foi a criação de uma ferramenta interativa que facilita a inserção de personagens virtuais autónomos em ambientes virtuais 3D. O caso de estudo usado foi o da cidade de Mértola por estar disponível um modelo 3D do terreno, das muralhas do castelo e de algumas casas do interior deste [3]. A Figura 4.1 ilustra uma cena gerada com a ferramenta desenvolvida, que se designa por Mértola Virtual.

Neste capítulo descrevem-se os detalhes da implementação da ferramenta, apresentam-se os principais desafios enfrentados e justificam-se as decisões que foram tomadas ao longo do seu desenvolvimento. Este desenvolvimento envolveu 4 fases: 1) testes preliminares com base na conversão direta do resultado do projeto anterior [3] e inserção de personagens virtuais no cenário, 2) conversão adequada do terreno de implantação do castelo e a ampliação do terreno ao redor deste castelo, 3) inserção de elementos adicionais e personagens autónomos no cenário e 4) criação da interface da aplicação. O software utilizado foi o Unity 3D e a programação dos *scripts* foi feita em C#, como justificado anteriormente.

Figura 4.1: Pormenor da simulação com personagens virtuais autónomos no mercado dentro das muralhas de Mértola.

4.1 Testes preliminares

A 1ª etapa do desenvolvimento deste projeto consistiu na identificação dos processos para a importação do modelo de Mértola do Alexandre Carvalho [3] criado com a ferramenta CityEngine, este software e o Unity3D são compatíveis entre si, pois é possível importar e exportar ficheiros no formato .FBX, a importação do modelo da cidade de Mértola gerada na plataforma CityEngine para o Unity3D realizou-se através deste formato.

A importação pode ser feita de maneira direta utilizando o *Assets* de importação padrão do Unity 3D (Menu: Assets > Import New Asset, escolhendo .FBX) ou arrastando para a área dos *Assests* do projeto.

Para testar o ambiente importado do CityEngine, foi necessário realizar a aplicação da operação *"generate collider"* no ficheiro .FBX importado para o Unity 3D, já que o primeiro teste que se pretendia fazer era colocar um personagem virtual a percorrer o ambiente dentro da muralha da cidade. Ao aplicarmos o "*generate collider"*, o Unity 3D aplica para toda a malha poligonal um *mesh collider*, evitando assim que o personagem venha a colidir com as paredes e outros objetos.

Na fase de importação para Unity3D, é selecionada automaticamente uma variação de fatores de escala entre os dois sistemas, é possível aplicar uma escala global ao modelo importado sempre que a escala do ficheiro original (do modelo) não se encaixar na escala pretendida no projeto. O sistema de física da plataforma Unity espera que 1 metro no mundo do jogo seja 1 unidade no ficheiro importado. Neste caso o *Scale Factor*[43] das *meshes* do projeto será de 1, que corresponde a 1 metro no ambiente Unity 3D. No caso do modelo importado, o Unity 3D interpretou o *Scale Factor* como 0.01 e foi necessário passar para o scale *factor* igual a 1.

A Figura 4.2 mostra o cenário do ambiente virtual da cidade de Mértola após a importação para a plataforma Unity 3D, assim como o fator de escala aplicado na importação (igual a 1). É possível notar uma cor escura atrás da cidade de Mértola, isso deve-se a facto de que é necessário acrescentar um céu. Este procedimento foi feito através da obtenção do *skybox* na Asset Store[44] do Unity 3D, onde existem outros modelos gratuitos. É possível também criar de raiz um *skybox*[45] dentro do ambiente Unity 3D.

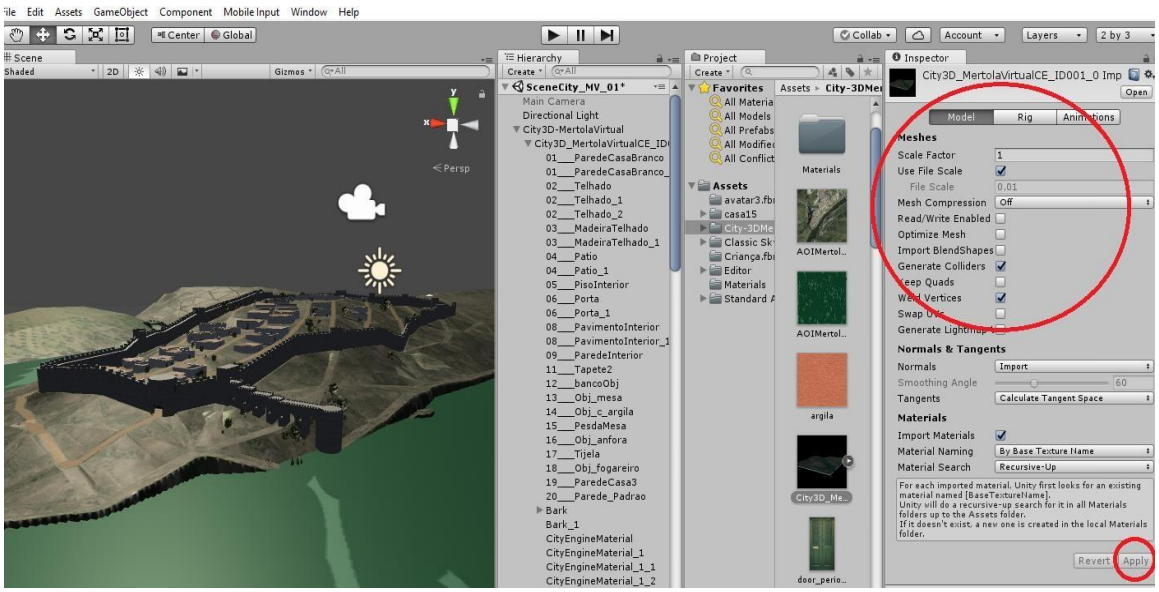


Figura 4.2: Cidade de Mértola importada para o Unity 3D.

O próximo passo foi fazer um pequeno teste com um personagem virtual no cenário importado para o Unity 3D, com intuito de testar se o ambiente estaria preparado para suportar interação com humanos virtuais em tempo real. Recorreu-se a recursos padrão do Unity 3D:

---

[43]Scale Factor*:* https://docs.unity3d.com/Manual/FBXImporter-Model.html

[44]Asset Store: https://assetstore.unity.com/packages/2d/textures-materials/sky/classic-skybox-24923

[45] Skybox: https://docs.unity3d.com/Manual/class-Skybox.html

- **Personagens virtuais**: Menu - Assets > Import Package > Characters;
- **Camera**: Menu - Assets > Import Package > Cameras.

Estes recursos padrão são gratuitos e estão disponíveis num repositório da plataforma Unity. Num primeiro teste usou-se somente um personagem interativo.

O resultado do teste foi satisfatório, pois foi conseguido atingir o objetivo proposto, que seria ter um personagem a percorrer o ambiente dentro da muralha em diferentes direções conforme a escolha do utilizador, para isto escolheu-se um ThirdPersonController (Ethan), com as opções Assets > Standard Assest > Characters > Prefabs > ThirdPersonController. O personagem virtual é dotado de 2 importantes *scripts* que o habilitam a andar e a ser controlado por teclado.

Para a realização do segundo teste, usou-se um personagem virtual do DAZ 3D que se importou para o Unity3D. Esta experiência foi necessária para que se observasse o comportamento de personagens virtuais oriundos de outras ferramentas, de forma a perceber quais as limitações e os possíveis desafios a enfrentar no decorrer do projeto, já que um dos principais objetivos seria a importação de vários personagens virtuais ao longo do projeto. Uma vez importado, criou-se um pequeno *script* somente para fazer com que um avatar seguisse outro avatar. O teste envolveu (Figura 4.3):

- 2 Personagens virtuais oriundos da plataforma DAZ 3D, e o recurso a um *script* para que um personagem pudesse seguir o outro.
- 3 clones do mesmo personagem do primeiro teste.

Figura 4.3: Humanos Virtuais no Cenário

Os testes apresentaram resultados positivos, porém, notou-se uma certa dificuldade dos personagens virtuais de seguirem uns aos outros nas partes mais íngremes do terreno, o que se deve ao cálculo do percurso. Notou-se também que em certos trechos da cidade os avatares caíam, apesar do terreno ter os respetivos *mesh colliders* gerados no momento da importação.

Ao importar a cidade de Mértola do CityEngine para o Unity, de forma automática o Unity como padrão interpreta tudo procedente da outra ferramenta como uma *mesh*, que até certo ponto é benéfico pelo peso ser menor em termos de vértices e triângulos. Porém possui uma

desvantagem enorme: não se poder ajustar a *mesh*, ou seja, pode-se alterar o valor da escala e altura, mas de uma maneira global e não local, o que em determinados momentos é essencial para poder ter o ambiente mais real quanto possível.

Uma das vantagens do Unity é possuir uma ferramenta para edição de um tipo de estrutura designada *terrain* que permite aos programadores criar paisagens ricas em detalhes de um modo simples, rápido e direto comparado com o uso de ferramentas de modelação 3D, tais como Blender e o 3D Studio Max. ATabela 5 identifica os pontos comuns e as diferenças em relação à utilização do mesh e *terrain*[46] no Unity 3D.

Tabela 5: Os pontos comuns e as diferenças entre *mesh* e *terrain* no Unity 3D.

Descrição	**Terrain do Unity 3D**	**Mesh do Unity 3D**
Permite o multi-texturing	Sim	Não
Utilização de heightmaps	Sim	Não
Modelação em outras ferramentas	Não	Sim
Menor peso em termo de vértices e triângulos	Não	Sim
Criação de cavernas	Não	Sim
Criação de LODs (níveis de detalhes)	Sim	Não
Ferramenta para a criação da vegetação e detalhes.	Sim	Não
Criação de colliders	Sim	Sim
Navegação (navmesh)	Sim	Sim

Cabe destacar que a utilização tanto de *mesh* ou de *terrain* depende muito do objetivo proposto, sendo necessário identificar que tipo de cenário é pretendido. A Tabela 6 exemplifica alguns tipos de cenários [47] e a recomendação da utilização.

Tabela 6 :Cenários recomendados.

Cenário	**Terrain do Unity 3D**	**Mesh do Unity 3D**
Floresta com muitas árvores, colinas íngremes, montanhas e rios.	Recomendável	Não recomendável
Uma prisão com corredores e pisos.	Não recomendável	Recomendável
Uma vila muito simples plana e com poucas casas	Não recomendável	Recomendável
Floresta integrada com conjunto de vilas medievais	Recomendável	Não Recomendável

[46] Pontos comuns e as diferenças entre *mesh* e *terrain*: https://forum.unity.com/threads/unity-terrain-vs-terrain-mesh-which-one-should-be-good-for-me.118366/

[47] Tabela de recomendação: https://www.quora.com/Should-I-use-a-terrain-or-a-mesh-to-create-the-floor-of-my-level-in-Unity

Cavernas com rochas pré-fabricadas	Não Recomendável	Recomendável
Cidade plana, com casas e edifícios	Não Recomendável	Recomendável

Como um dos objetivos do nosso projeto era ter um ambiente o mais real quanto possível identificou-se a necessidade de se converter a *mesh* para *terrain.* Com base nos aspetos referidos nas tabelas 5 e 6.

4.2 Terreno

Na versão final do modelo da cidade de Mértola realizado pelo Alexandre Carvalho [3], o foco principal era a cidade em si, sendo o terreno aproximadamente um quadrado de 1 km de lado, uma área adequada tendo em conta a dimensão da cidade na Idade Média. Ao importar este modelo para o Unity 3D, foram observadas duas características que quisemos melhorar: por um lado, a malha do terreno (*mesh*) tinha demasiados polígonos e, por outro lado, a distância de 1 km era insuficiente para obter o nível de realismo necessário para a simulação pretendida. Na realidade, na perspetiva do utilizador era possível com muita facilidade ver a linha do fim do mundo, como se pode ver na Figura 4.4, tendo surgido assim o desafio de agregar uma área de terreno envolvente de modo a minorar este efeito.

Figura 4.4: Linha do fim do mundo.

4.2.1 Conversão do terreno

Ao importar este modelo para o Unity 3D verificou-se que a malha do terreno tinha demasiados polígonos como já referido. Esta característica levanta problemas, nomeadamente, o facto de ser difícil editar a malha (processo que tem de ser efetuado na fase de ampliação do terreno, como explicado na secção seguinte) e a dificuldade de recorrer a uma funcionalidade do Unity, designada por *navmesh*, que facilita a inserção de personagens autónomos animados no cenário.

Uma vez identificada a necessidade de se converter a malha do terreno para *terrain* foi necessário fazer um estudo para identificar, como seria o processo de conversão, já que o mesmo não é automático na plataforma Unity 3D. Foi usado na comunidade um *script* chamado

object2terrain[48] que realizava essa operação. Para isso bastava fazer o download e importá-lo para os *Assets* do projeto.

Depois do *script* estar incorporado na plataforma Unity 3D, o próximo passo, foi preparar o ambiente para a conversão. Foi então necessário separar as casas e a muralha do terreno, deixando somente a textura do terreno como referência, para que posteriormente todas as casas fossem colocadas no mesmo lugar (Figura 4.5).

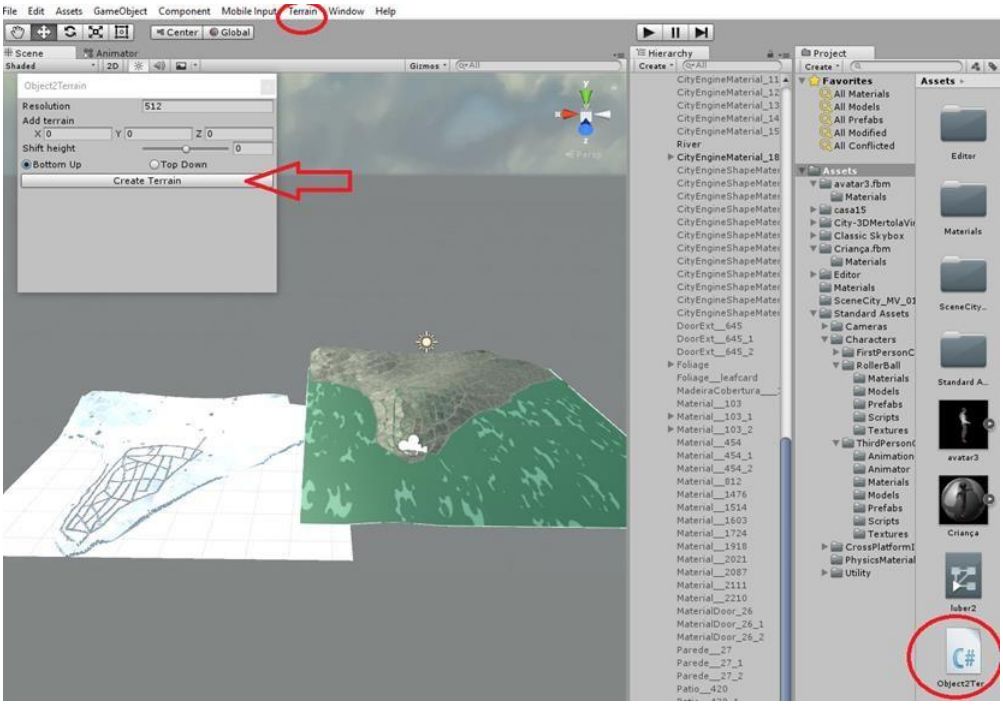


Figura 4.5: Conversão da malha do terreno para *terrain*.

Uma vez concluída a conversão do terreno, as casas e muralha foram recolocadas nos respetivos lugares, sendo necessário realizar pequenos ajustes para que as casas encaixassem no terreno. Estes ajustes foram feitos no editor de terrenos do Unity 3D (ver anexo A).

4.2.2 Ampliação do Terreno

Para proceder à ampliação do terreno em torno da área do castelo, foi necessário identificar o melhor processo para a sua obtenção.

A primeira alternativa para eliminar o efeito de fim do mundo seria aumentar o terreno no CityEngine. Porém esta alternativa foi logo descartada pois era necessário rever todas a medidas que foram usadas no processo de conceção da cidade de Mértola.

A segunda alternativa seria utilizar uma semiesfera (abóboda[49]), ou seja, uma cobertura sobre a cidade que representasse o céu. Recorremos ao Blender, para criar a cobertura e posteriormente exportou-se para o Unity no formato .FBX. A grande vantagem desta alternativa seria que neste caso podemos limitar o terreno de navegação dos personagens virtuais através de um *mesh collider*, aplicado na abóboda, assim como atribuir uma textura aceitável que representasse bem o céu. Porém, como se pode observar na parte direita da Figura 4.6, ainda teríamos um pedaço de terreno visível. Nota-se também a perda do *skybox*, ou seja, o conjunto de

[48] Script "object to terrain" retirado de: http://wiki.unity3d.com/index.php?title=Object2Terrain

[49] Abóboda: https://www.infopedia.pt/dicionarios/lingua-portuguesa/ab%C3%B3bada

texturas aplicadas no céu, porém o problema poderia ser contornado com uma imagem mais nítida e com uma iluminação mais adequada. Por estes motivos a alternativa da abóboda foi descartada.

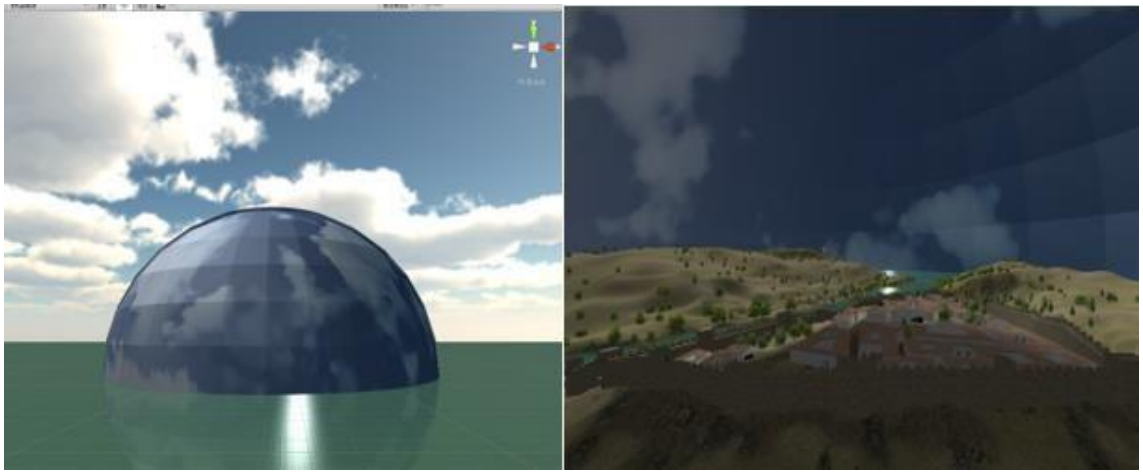


Figura 4.6: Abóbada sobre a cidade de Mértola

A terceira alternativa, que foi adotada, foi aumentar o terreno em torno do já existente. Para isto foi necessário rever as coordenadas em que o projeto inicial foi baseado. A forma mais viável e confiável seria obter os dados no *Google Earth*[50], para assim podermos utilizar o mesmo padrão tanto nas coordenadas como no tamanho. Essa alternativa foi escolhida já que de acordo com o estudo de ferramentas realizado previamente, a ferramenta Google SketchUp[51] é completamente compatível com Unity 3D, sendo possível exportar ficheiros .FBX. Outra vantagem seria referente à geolocalização do ficheiro, pois permite posicionar o projeto com precisão de latitudes e longitudes.

A Figura 4.7 mostra o pipeline seguido para a geração do terreno "extra" com coordenadas reais da cidade de Mértola. Os passos são detalhados em seguida.

Figura 4.7: Pipeline para a criação de terreno extra.

Geração de terreno através da ferramenta Sketchup Pro

De acordo com o Projeto inicial [3], as coordenadas que delimitavam o terreno eram: (37.633°, -7.668°); (37.633°, -7.656°); (37.642°, -7.668°); (37.633°, -7.642°). A partir destes valores calcularam-se as coordenadas adjacentes que posteriormente foram utilizadas para obter o lugar exato de onde foram extraídos os terrenos. Foi estabelecido que a distância das coordenadas dos pontos seria de aproximadamente 1 km.

50 Google earth: https://support.google.com/earth/answer/21955?hl=en

51 Google SketchUp: https://docs.unity3d.com/560/Documentation/Manual/HOWTO-ImportObjectSketchUp.html

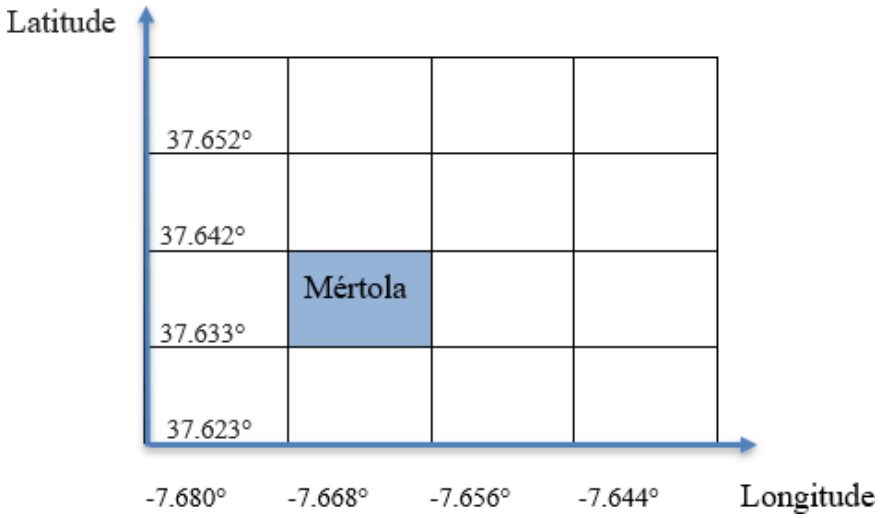


Figura 4.8: Quadrícula com as respetivas coordenadas que delimitam o terreno da cidade de Mértola.

A Figura 4.8, mostra uma quadrícula com as coordenadas (longitude e latitude) que delimitam o terreno da cidade de Mértola, para verificar a veracidade das mesmas. As coordenadas foram introduzidas no Google Earth e a Figura 4.9 mostra os respetivos pontos, representados pelos marcadores. O marcador 8.1 na Figura 4.9 por exemplo, representa as coordenadas (37.623°, -7.644°).

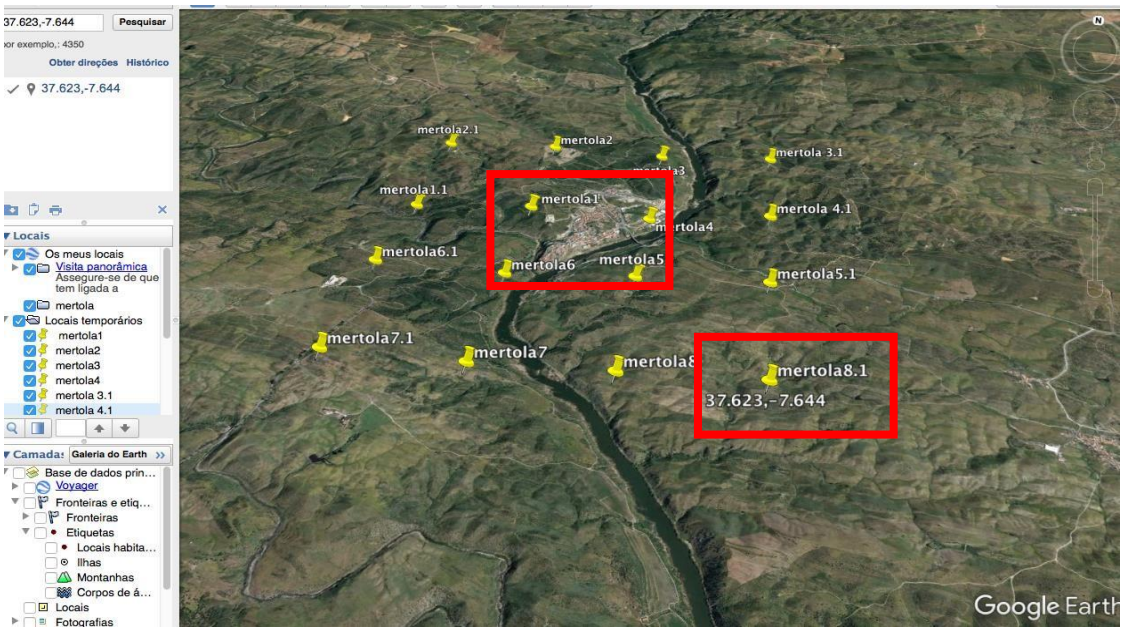


Figura 4.9: Coordenadas dos pontos da região com a aproximadamente 1 km de distância entre elas.

Uma vez verificado as coordenadas no Google Earth, o próximo passo foi utilizar a ferramenta Google Sketchup.

Foi identificado o terreno de Mértola destacado em vermelho no centro da Figura 4.9. O seguinte passo foi identificar a parte que seria adicionada ao terreno. Foram pensadas 2 alternativas:

I. Identificar e extrair somente a parte do terreno que dá continuidade ao rio. Esta alternativa seria muito viável pois pouparia recursos de processamento, pois teríamos uma *mesh* bem menos pesada.

II. Identificar e extrair terreno em volta da cidade de Mértola. Esta alternativa seria, num primeiro momento menos viável, pois quanto maior for a quantidade de terrenos maior e mais pesada será a *mesh*.

Foram realizados 2 testes, relativos aos pontos I e II mencionados anteriormente, e foi concluído que a diferença entre termos 4 *meshes* correspondentes à parte que da continuidade ao rio face as 8 *meshes* que correspondem a todo terreno do entorno, é pequena, e por isso optou-se pela segunda alternativa.

Uma vez escolhida quantas *meshes* teríamos para compor o terreno foram então executados os seguintes passos explicados detalhadamente no Anexo A.6:

1. O primeiro passo foi introduzir cada coordenada (longitude, latitude) no Google SketchUp, sendo uma de cada vez relativa ao ponto de interseção, e logo depois fazer a extração do terreno.
2. Transformar o terreno plano obtido num *terrain* com elevação.

A Figura 4.10 mostra o processo de montagem dos terrenos das várias regiões envolventes.

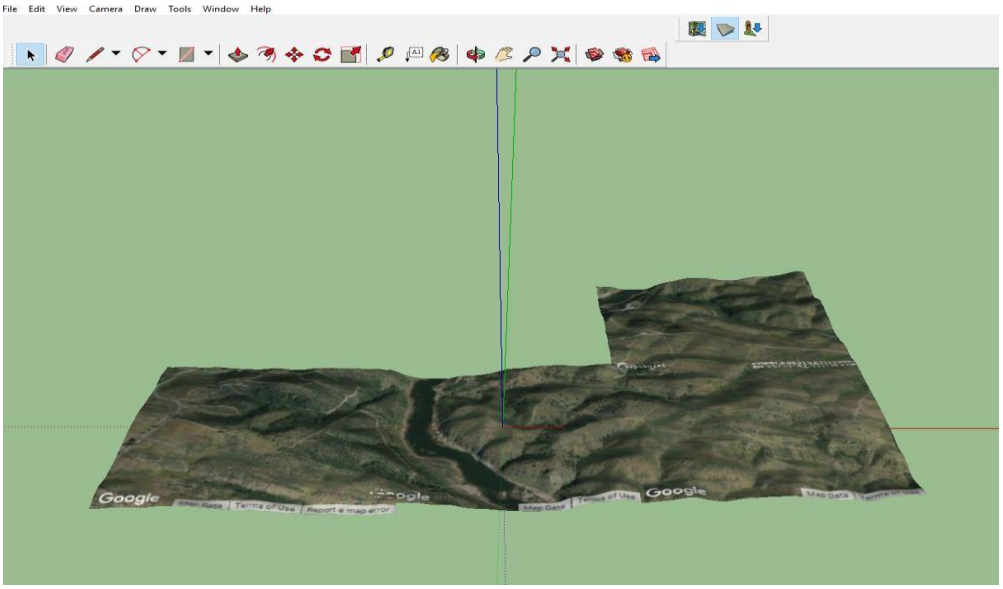


Figura 4.10: Regiões escolhidas sendo montadas.

Depois de repetido o processo de escolha para as 8 regiões envolventes do terreno inicial, foi obtido a região mostrado na Figura 4.11 .

Figura 4.11: Junção final das 8 regiões escolhidas.

Uma vez obtido o terreno final desejado, o próximo passo foi exportá-lo para o Unity 3D. A geração do terreno ampliado com a qualidade desejada foi um processo moroso que está detalhado no Anexo A.6.

A Figura 4.12 mostra o modelo da cidade de Mértola e a paisagem envolvente.

Figura 4.12: Cidade de Mértola integrada ao entorno *mesh* (parte vermelha).

As regiões envolventes podem estar representadas de 2 formas diferentes: *mesh* ou *terrain.* Procedeu-se a um teste de desempenho no âmbito de comparar o número de (vértices e triângulos das 2 versões). Para o referido teste foi levado em consideração as *directional lights*[52] (luzes direcionais), que são muito úteis para criar efeitos como a luz do sol nas cenas. As tabelas 7 e 8 apresentam a comparação entre os vértices e triângulos de acordo com a *directional light*.

Tabela 7: Tabela comparativa dos vértices e triângulos com luz direcional.

Com *directional light*	Vértices	Triângulos
Mesh (região externa)	47.5K	81.1K
Terrain (região externa convertida)	170.4K	291.7K

Tabela 8: Tabela comparativa dos vértices e triângulos sem luz direcional.

Sem *directional light*	Vértices	Triângulos
Mesh (região externa)	26.3K	41.4K
Terrain (região externa convertida)	87.7K	146.7K

[52] Directional light: https://docs.unity3d.com/Manual/Lighting.html

Em termos de comparação sobre o resultado obtido nas tabelas 7 e 8, era o que se esperava, quando comparado os vértices e triângulos com o *directional ligth* em relação à cena sem o *directional ligth*, o número de vértices e triângulos praticamente duplica, isso deve-se ao facto de ao incindir a luz sobre a *mesh* e o *terrain* geram-se sombras projetadas, que são as responsáveis pelo aumento significativo dos vértices assim como dos triângulos.

Comparando *mesh* e *terrain*, observa-se que a *mesh* (região externa) tem um número de vértices e triângulos muito inferior em relação ao *terrain* (região externa convertida), constituindo assim uma abordagem mais leve em termos de processamento. melhor desempenho face ao *terrain*. Porém, apesar de termos um melhor desempenho, para garantir um resultado mais harmonioso e correto relativamente às coordenadas geográficas, optámos pelo tipo *terrain*.

Em termos de projeto futuro, por exemplo, se for necessário acrescentar mais terreno à volta, numa escala global o *terrain* apesar de ser computacionalmente mais "pesado" tem a vantagem de ser mais realista por poder utilizar as ferramentas incorporadas no Unity. A adição de terreno será mais viável, devido à possibilidade de utilização da função *terrain.setneighbors*[53] que estabelece a conexão dos terrenos vizinhos, garantindo a "costura" e encaixe dos LODS *(Level of Details*) do terreno. No Anexo A.7 é explicado e exemplificada a utilização deste recurso.

Correção da junção dos terrains

Uma vez convertidos, ajustados e encaixados os *terrains* da cidade de Mértola foi preciso eliminar alguns *gaps* existentes nas zonas de junção dos terrenos, um efeito indesejável visualmente notório, dependendo da posição do utilizador.

Para se poder aplicar o *script* s*etneighbors* (descrito no Anexo A.7) é necessário corrigir os *terrains* devido as diferenças de alturas entre estes.

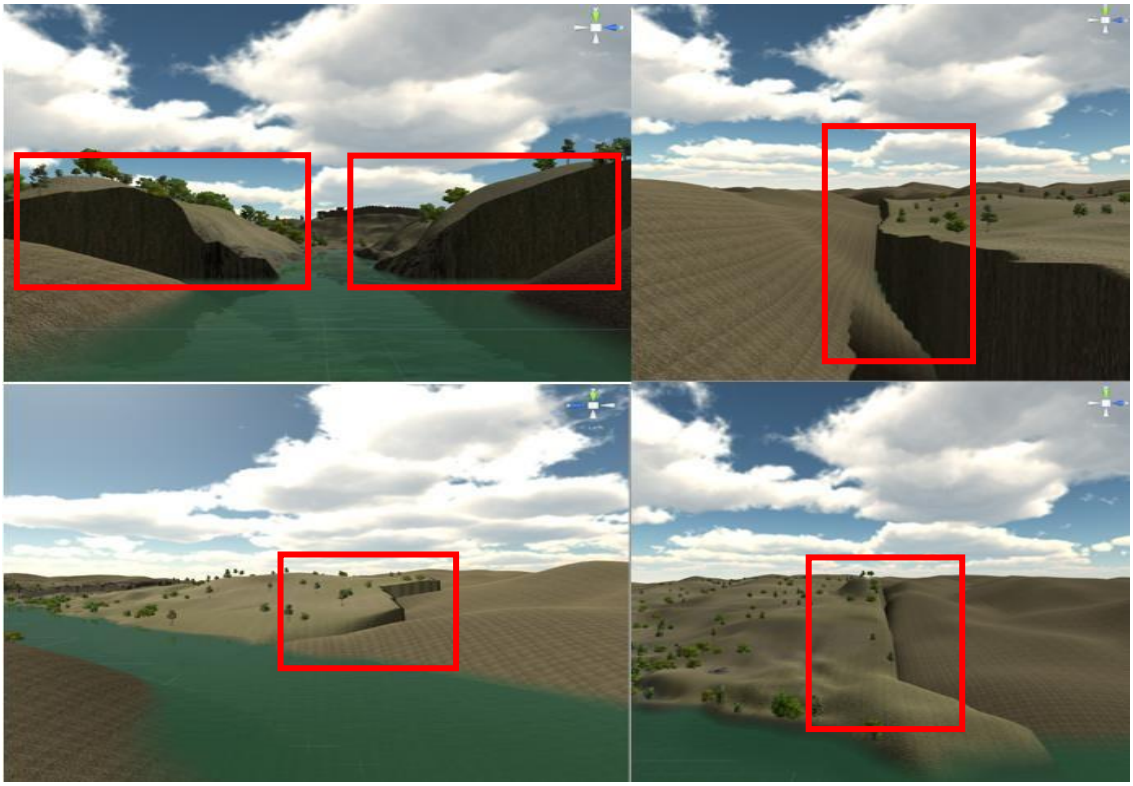


Figura 4.13: *Gaps* na junção dos terrenos.

[53] Terrain.setneighbors: https://docs.unity3d.com/ScriptReference/Terrain.SetNeighbors.html

Na Figura 4.13 nota-se que apesar de termos feito o encaixe ainda faltava resolver alguns *gaps* gerados na adição dos terrenos. Existem várias formas de eliminar os *gaps*, em particular existem várias ferramentas pagas na *assets store* do Unity para o efeito.

Porém existem também outras soluções sem nenhum custo, tirando proveito das próprias ferramentas do Unity, modificando as alturas das extremidades, porque o *setneighbors* por si só não trata as diferenças entre elas. Assim sendo, foram testadas as duas abordagens para saber qual seria a mais viável.

A primeira testada foi a *terrain stitcher*[54] adquirida da asset store do Unity 3D. A ferramenta demostrou ser muito lenta, no processo de costura apesar de já tratar as diferentes alturas e corrigir pequenos *gaps* no terreno. No nosso modelo da cidade de Mértola em particular, não funcionou, devido ao tamanho e complexidade do terreno. A ferramenta mostrou-se funcional apenas em terrenos com menos "carga".

Com o objetivo de maximizar aplicação minimizando os custos, o próprio *terrain* tem a particularidade de resolver este problema com as ferramentas de edição, sem nenhum custo associado, mediante isso o terreno foi cuidadosamente "costurado".

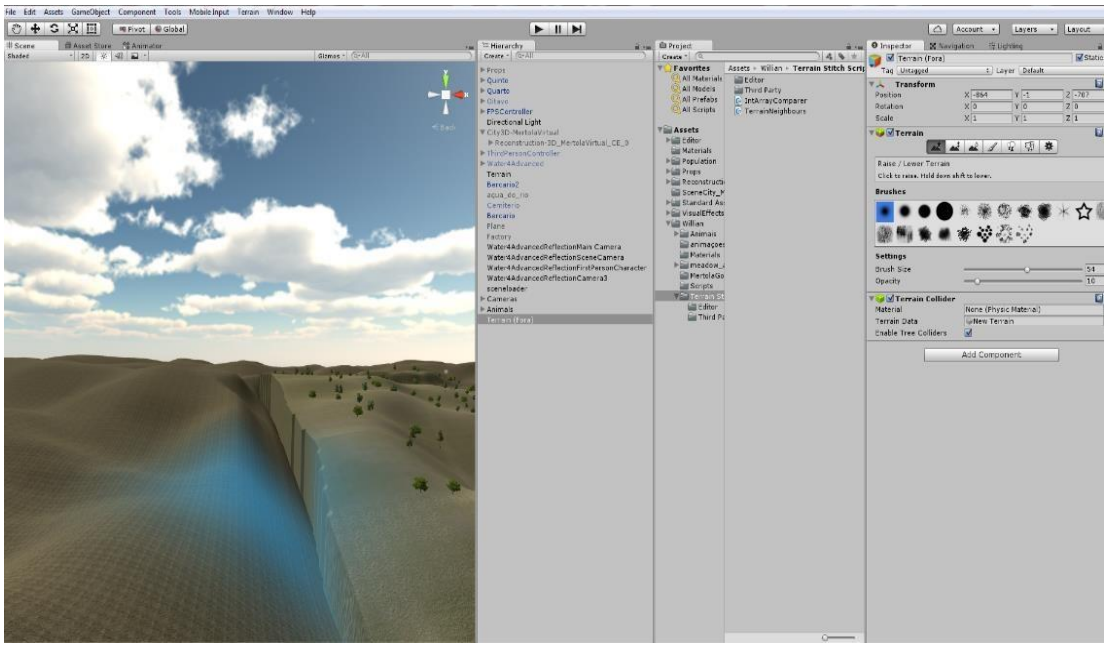


Figura 4.14: Exemplo de um *gap* a ser testado com a ferramenta.

A Figura 4.14 mostra o *terrain* antes da aplicação da ferramenta de edição, nota-se um parte mais azulada, nela será onde haverá uma leve alteração no intuito de poder nivelar os *terrains*. A sua aplicação é bastante intuitiva, somente é necessário escolher o tamanho do pincel desejado assim como área, neste procedimento as tarefas são manuais levando em consideração a altura em que se deseja chegar.

Já depois de modelado nota-se a resolução dos problemas dos *gaps*, o que mostra mais uma vez a vantagem de converter uma *mesh* em *terrain*. Na Figura 4.15 foram colocadas diferentes texturas no intuito de se testar a viabilidade da ferramenta, sendo mais fácil de localizar onde os *gaps* se encontram.

54 Terrain stitcher: https://assetstore.unity.com/packages/tools/terrain/terrain-stitcher-42671

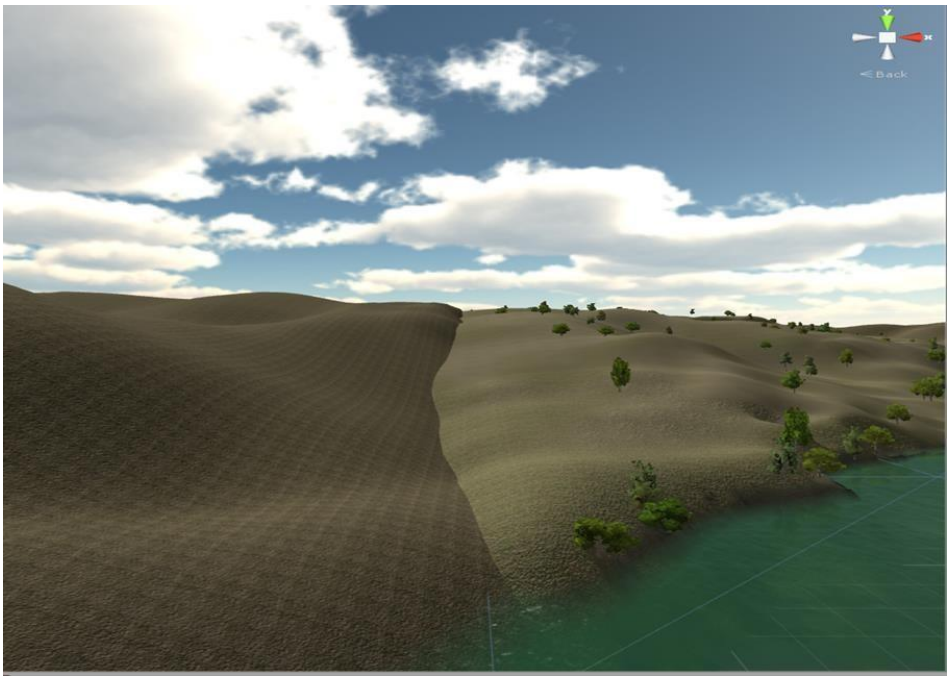


Figura 4.15: Terrenos unidos sem as diferenças de alturas.

Com aplicação da mesma textura em ambos terrenos não é possível notar que os terrenos foram "costurados", garantindo assim a originalidade do terreno no ponto de vista do utilizador.

Uma vez terminada a preparação do terreno da cidade de Mértola através da conversão, adição de terreno extra e sua posterior junção foram inseridas casas e a muralha, melhorou-se a textura do terreno e incluiu-se vegetação.

O terreno da cidade de Mértola ficou assim pronto para a inserção dos personagens e de outros elementos virtuais. Na próxima secção serão explicados: a inserção dos objetos medievais da época no cenário virtual; a criação dos humanos virtuais característicos da época e as suas animações; a importação dos animais virtuais da época e os seus comportamentos.

Construção de terreno real no Unity3D

O modelo de um terreno pode ser gerado de várias formas, e para isso são necessários algumas ferramentas e metodologias para a criação e importação dos mesmos. O Unity 3D tem um sistema padrão de terrenos que permite a criação de vastos ambientes abertos como por exemplo: montanhas, rios, ilhas, florestas, entre outros, no entanto para que um modelo de terreno seja uma aproximação realista é necessário introduzir coordenadas reais.

Um modelo terreno realista é gerado por ferramentas específicas e posteriormente importado para o Unity 3D. Nestas ferramentas um terreno que poderá ser gerado numa das duas formas: como uma *mesh* ou como um *heightmap*, dependendo da ferramenta utilizada.

No anexo A.1 encontra-se uma lista de várias ferramentas pesquisadas que existem para criação de terrenos, estas ferramentas funcionam de uma maneira análoga: um terreno é criado a partir de coordenadas reais e posteriormente é exportado como uma imagem bidimensional utilizada para armazenar valores relativos à elevação de uma superfície para exibição como gráficos tridimensionais, sendo denominada *heightmap*[55] ou *heightfield* (*"*mapa de altura").

[55] Heightmap: https://en.wikipedia.org/wiki/Heightmap

A imagem que armazena um *heightmap* possui apenas um canal que é interpretado como a distância de um determinado ponto até o chão. Geralmente as imagens apresentam-se em escala de cinza. O preto representa a menor e o branco a maior altitude possível. Um *heightmap* pode ser de 8-bits e representar somente alturas de 0 a 256 ou de 16-bits e representar alturas com valores de 0 a 65536.

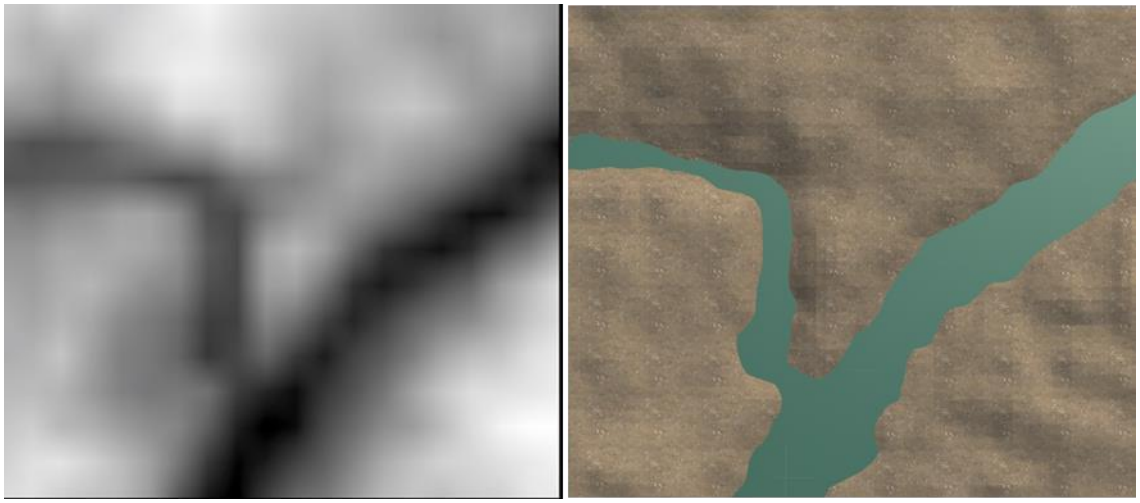


Figura 4.16: À direita temos o terreno convertido no Unity 3D a partir do heightmap mostrado a esquerda

A Figura 4.16 mostra à direita o terreno de Mértola importado e convertido no Unity 3D, onde foi criado a partir do *heightmap* gerado pela ferramenta CityEngine apresentado à esquerda da figura.

As etapas subsequentes da criação de um *heightmap* e importação para Unity 3D serão descritas nas próximas secções.

Um terreno real também pode ser criado como uma malha poligonal. A Figura 4.17 representa uma malha poligonal (*mesh*) com sua respetiva textura importada no Unity 3D, sendo possível observar os polígonos que formam a referida *mesh*.

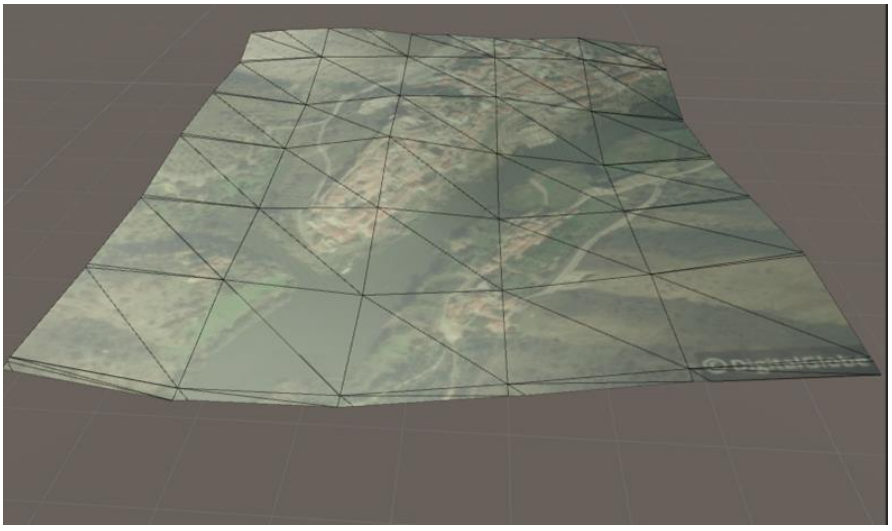


Figura 4.17: Uma *mesh* importada para o Unity 3D.

A escolha de uma ferramenta para gerar um terreno real é importante e assenta em fatores que são determinantes. Embora todas as ferramentas aqui mencionadas permitam criar terrenos,

há especificidades próprias que podem adequar-se com maior ou menor eficiência ao projeto em si.

A Figura 4.18 demonstra que usando coordenadas reais é possível criar terrenos reais através de ferramentas de software de criação e serem importadas para o Unity 3D de duas maneiras diferentes.

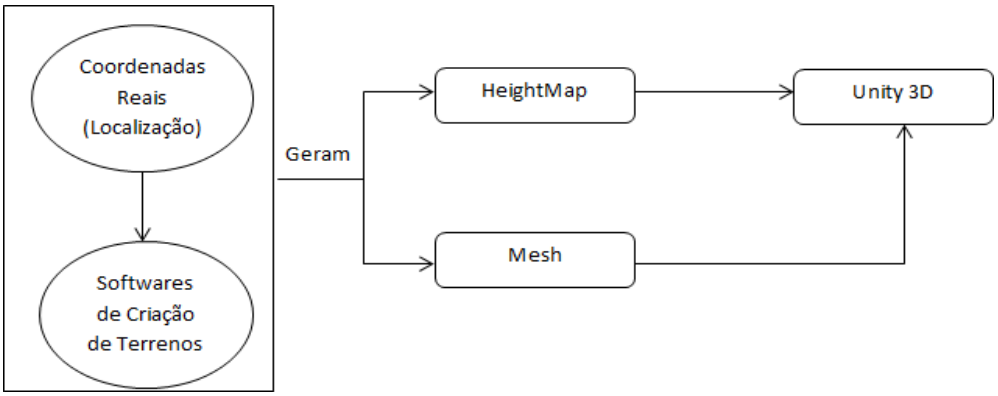


Figura 4.18: Diagrama que ilustra as formas de gerar um terreno

As principais ferramentas encontradas foram analisadas através de vários parâmetros, sendo os requisitos obrigatórios:

- Existir uma versão gratuita ou gratuita por tempo limitado;
- Ser compatível com Unity 3D;
- Ter capacidade de gerar *heightmaps* ou *mesh*;
- Possuir documentação para auxílio e vasta comunidade de utilizadores.

Deste modo, foram selecionadas duas ferramentas que ilustram as duas principais formas de gerar terrenos reais: o Sketchup Pro que, por norma, gera terrenos e os exporta como uma *mesh*, e o terrain.party que gera um *heightmap* a partir da introdução da localização desejada.

A criação de terreno através da ferramenta Sketchup Pro foi descrita ao pormenor na secção 4.2.2, e nas secções seguintes serão explicadas como criar um terreno real através de um *heightmap* gerado pela ferramenta escolhida terrain.party, assim como as diferentes maneiras de importá-lo para o Unity 3D.

Como criar um *heightmap* de um terreno real

Para criar um *heightmap* a partir de coordenadas reais utilizaremos o terrain.party, que é uma ferramenta totalmente gratuita e intuitiva que simplifica o método de gerar *heightmaps* de qualquer parte do mundo. O terrain.party funciona extraindo os dados da zona de interesse e gerando um ficheiro .png com a respectiva zona.

O terrain.party irá gerar um ficheiro .zip, contendo vários *heigthmaps* em formato .png gerados a partir dos dados captados por diferentes radares, neste caso já relativos a Mértola:

- **Mertola Height Map *(ASTER 30m)*[56].png**: É um conjunto de dados de aproximadamente 30m de resolução, cobrindo a terra para altas latitudes, porém com algumas lacunas ocasionais, devido à alta concentração de nuvens ou montanhas. Projetado para capturar imagens de alta resolução da Terra, *the Advanced Spaceborne Thermal Emission and Reflection Radiometer* (ASTER) é um

[56] ASTER 30m*:* http://asterweb.jpl.nasa.gov/gdem.asp

dos cinco instrumentos a bordo do satélite Terra da NASA, onde os seus dados são utilizados para criar mapas detalhados da temperatura da superfície terrestre, emissividade, reflectância e elevação na terra.

- **Mertola Height Map (SRTM3 v4.1)57.png**: Os dados foram fornecidos originalmente pela missão *Shuttle Radar Topography (*SRTM), porém o SRTM3 v4.1, tem uma resolução de aproximadamente 30m nos Estados Unidos e nos outros lugares de aproximadamente de 90m.
- **Mertola Height Map *(SRTM30 Plus)*[58].png**: É um conjunto de dados derivados do SRTM de aproximadamente 900m de resolução, incluindo dados batimétricos.
- **Mertola Height Map (Merged).png** : Fornece uma combinação dos dados do *ASTER 30m, USGS NED 10m*[59] que é um conjunto de dados de aproximadamente 10m de resolução que cobre os Estados Unidos e o *SRTM30 Plus,* com isso viabiliza bons dados globais de elevação *(ASTER)* com melhores dados de elevação dos Estados Unidos (*USGS NED)* e com um excelente preenchimentos de buracos e batimetria do *(SRTM30 Plus).*

Para um *heightmap* mais preciso, é necessário ter uma maior quantidade de dados possíveis e o Mertola Height Map (Merged).png fornece isso, uma combinação de dados derivados de vários radares, tornando-se assim o melhor candidato para a escolha.

Todos os *heightmaps* por norma são gerados em ficheiros de imagens, normalmente .png, porém para a sua exportação especialmente para o Unity 3D é necessário redimensionar a imagem para 1024 x1024 pixels, que é medida padrão aceite pelo Unity 3D. Como o terrain.party fornece imagens a 1081 x1081 pixels, para fazer essa conversão é necessário utilizar uma ferramenta de edição de imagens. No nosso caso usou-se a ferramenta gratuita Gimp.

Importar um *heightmap* para Unity 3D

O Unity 3D é capaz de gerar um terreno a partir de um *heightmap*, porém somente em ficheros do tipo ". raw". Mas quase todas as ferramentas de criação de *heightmaps* geram somente ficheiros em formato de imagem de formato .png ou .jpeg. Existem duas soluções para resolver este problema: A primeira segue a abordagem análoga a descrita por Saldaña e Johanson [53] que exporta o terreno em formato de imagem .png e é convertido para o formato .raw file usando o Photoshop.

Para poder seguir o processo descrito por Saldaña e Johanson [53], em primeiro lugar é necessário conhecer o editor de terrenos do Unity 3D.

O editor de terrenos do Unity 3D possui um conjunto completo de ferramentas para tornar o processo de criação de terrenos simples e rápido. Porém, para a sua correta utilização, será necessário importar alguns elementos importantes para que o terreno seja o mais real possível: texturas, árvores e água entre outros. Esses elementos podem ser criados em ferramentas externas ou, como no caso das árvores, é possível criar através do editor de árvores. O Unity 3D inclui, o *package environment,* com elementos para o ambiente que é preciso importar de modo que se descreve:

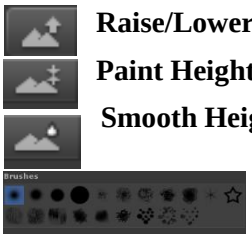
- **Adicionar package environment**: Assets>Import Package>Environment
- **Adicionar terreno**: GameObject > 3D Object > Terrain

57 SRTM3 v4.1: http://www.cgiar-csi.org/data/srtm-90m-digital-elevation-database-v4-1

58 SRTM30 Plus: http://topex.ucsd.edu/WWW_html/srtm30_plus.html

59 USGS NED 10m: https://nationalmap.gov/elevation.html

- **Ferramentas de altura**: (Figura 4.12)



Raise/Lower Height: Subir e Baixar altura;

Paint Height: Pintar altura;

Smooth Height: Suavizar altura;

Brushes: Escolha do pincel.

Depois de convertido o ficheiro para o formato .raw, será necessário importá-lo para o Unity 3D, para isso criamos um novo projeto (é recomendável guardar a cena que foi criada automaticamente ao originar um novo projeto).

Neste projeto inserimos um objeto do tipo *terrain* na nossa cena para poder importar o *heightmap* (GameObject > 3D Object > Terrain) ou (Hierarchy > Create > 3D Object > Terrain). Uma vez criado é possível ver na aba Hierarchy a árvore da cena.

O tamanho do *terrain* deve ser ajustado. Para isso é necessário o inspector e aceder às configurações (Figura 4.19). As dimensões devem corresponder ao tamanho que foi dado ao *heightmap* na sua criação. Quando um objeto do tipo terreno é criado, o Unity 3D por omissão define os tamanhos de 500x500 e altura 600. No caso do *heightmap* de Mértola os parâmetros utilizados foram 1024x1024 e 110 de altura.

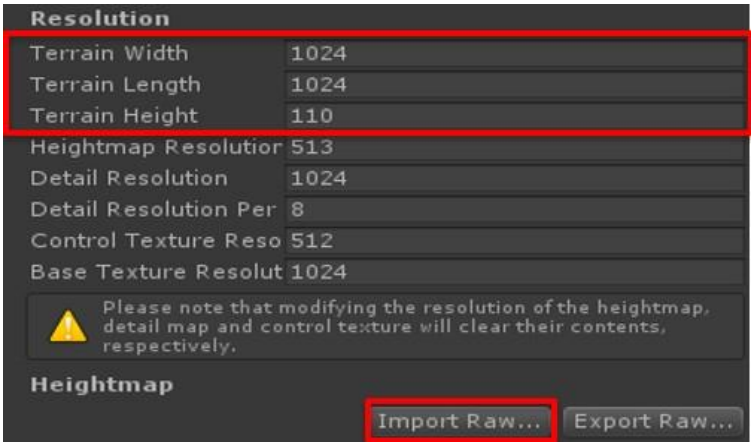


Figura 4.19: Ajustes do tamanho do terreno para importação do *heightmap*.

Após definir as dimensões, já se pode importar o *heightmap* através do botão "Import Raw". Uma vez selecionado, surge uma janela exibindo vários dados relativos ao mesmo aparecerá, na qual será necessário verificar alguns campos tais como:

- Se a profundidade (*Depth*) está em Bit16, podendo ser escolhido entre Bit8 e Bit16;
- Se as dimensões (*width e height*) estão corretas;
- E se o Byte Order esta na opção Windows, podendo ser escolhido tanto com Windows ou Mac.

Depois de ajustado corretamente, o *heightmap* será importado e teremos um terreno sem texturas. Com as ferramentas do editor de terrenos do Unity 3D acrescentam-se, texturas, árvores e efetuam-se ajustes pontuais (Assets>Import Package>Environment). É também possível recorrer à asset store do Unity 3D. A Figura 4.20 mostra o exemplo de um *heightmap* de Mértola importado para o Unity 3D, ainda sem texturas e com a textura aplicada posteriormente.

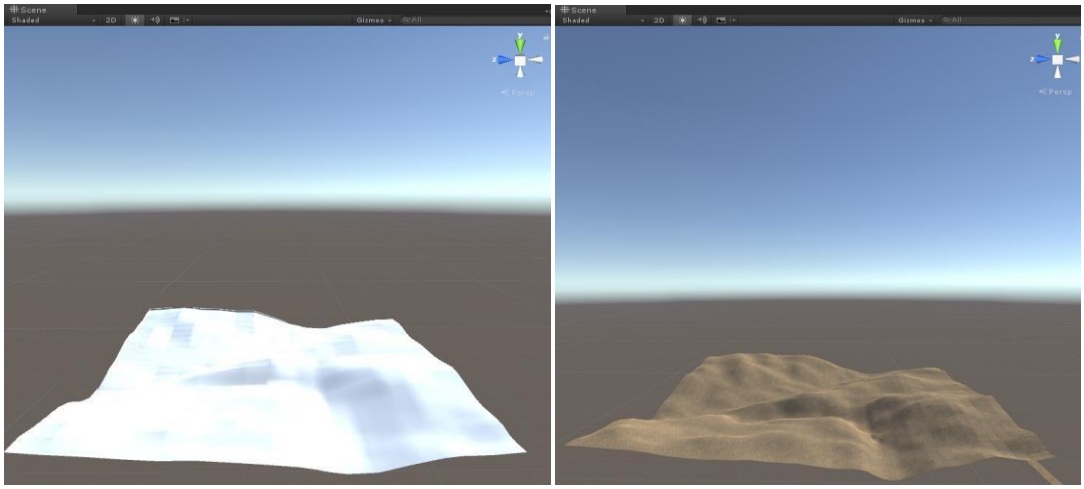


Figura 4.20: Antes e depois de texturizar o terreno.

Importar um *heightmap* para Unity 3D através de script

O segundo processo que foi testado para importar um *heightmap* recorreu ao *script* "HeightmapFromTexture", que pode ser obtido na wiki[60] do Unity 3D. Com esta solução um *heightmap* gerado por qualquer ferramenta num ficheiro .png pode ser importado para o Unity 3D sem a necessidade de se converter para o formato .raw que é o padrão para importações de *heightmaps.*

A resolução vertical da imagem será limitada é 8 bits, e o requisito mínimo para utilização do *script* é a versão 2.6 do Unity.

Podem também ser usadas texturas com cores, e neste caso as cores são avaliadas pelo seu valor na escala de cinza. Se a textura for de um tamanho diferente da resolução do terreno ativo, ela será redimensionada para se ajustar, aplicando o método *Nearest Neighbor scaling*[61] se a textura estiver configurada para não ter filtragem; caso contrário, será escalada utilizando o *bilinear filtering*[62].

A Figura 4.21 exemplifica o *heightmap* de Mértola gerado pelo CityEngine e importado para o Unity 3D através do *script* "*HeightmapFromTexture".*

Este *script* funciona de uma maneira relativamente simples, permite criar um terreno no Unity 3D a partir de um *heightmap* sem a necessidade de utilizar uma ferramenta para converter o *heightmap* em ficheiros. raw, sendo assim é possível poupar recursos e tempo.

60 Wiki do Unity 3d: http://wiki.unity3d.com/index.php/HeightmapFromTexture

61 Nearest Neighbor scaling: http://tech-algorithm.com/articles/nearest-neighbor-image-scaling/

62 Bilinear filtering: https://en.wikipedia.org/wiki/Bilinear_filtering

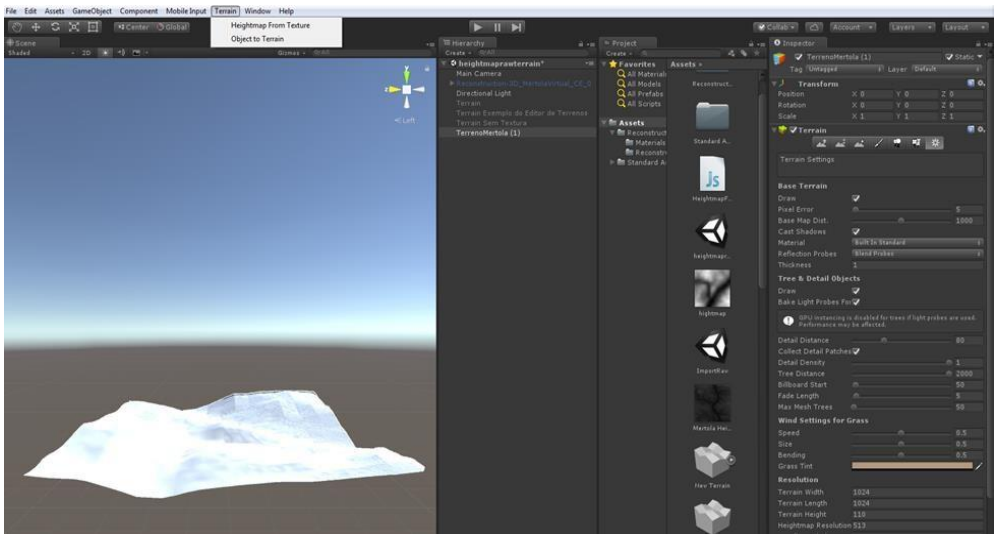


Figura 4.21: Importação de um *heightmap* através de um *script*.

A Figura 4.22 ilustra as etapas e as ferramentas testadas para a criação e conversão de terreno para Unity 3D.

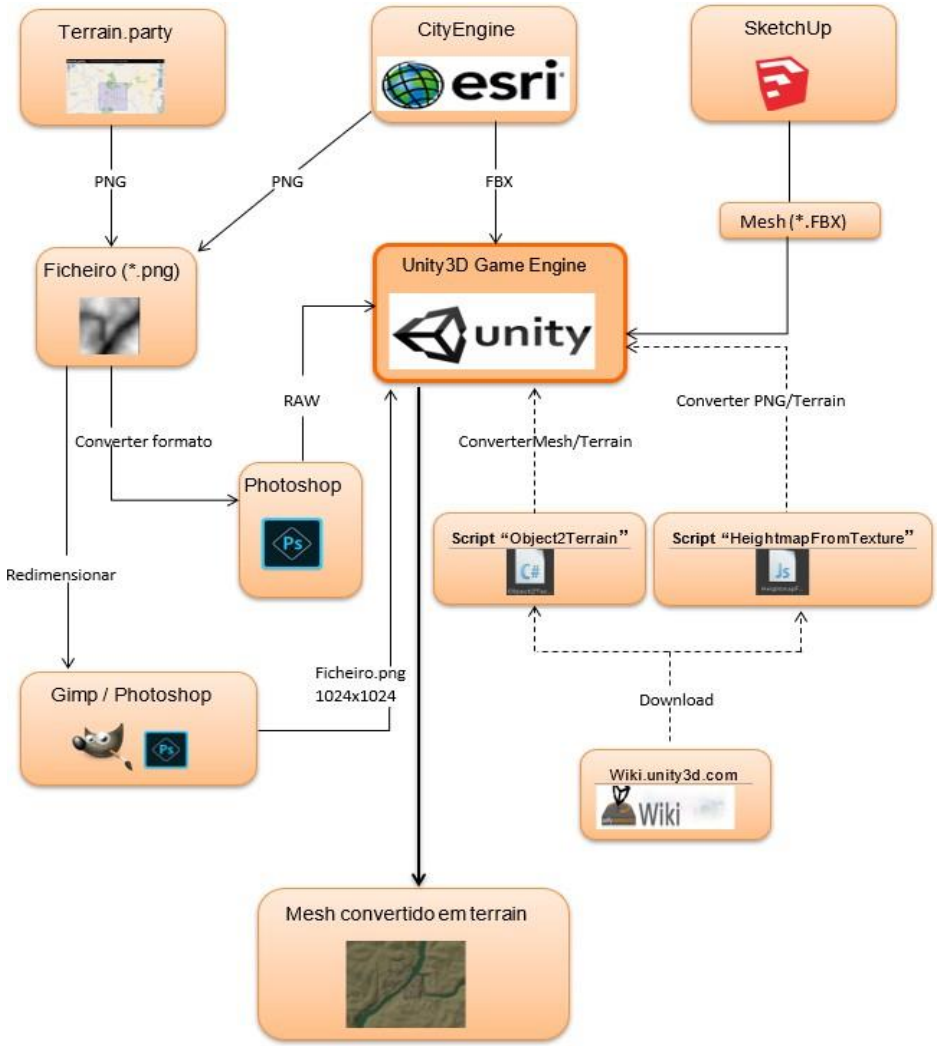


Figura 4.22: Etapas para a criação, conversão e expansão do terreno para Unity 3D.

Nesta secção foram descritos vários passos importantes para a concretização da ferramenta Mértola Virtual. Uma das premissas necessárias para a criação da ferramenta Mértola Virtual era conhecer as diferentes maneiras de obter um terreno real através de ferramentas específicas que fossem compatíveis com o Unity 3D. Fez-se uma pesquisa para obter as principais ferramentas e como resultado, foi definido o *pipeline* da Figura 4.22.

4.3 Inserção de elementos no cenário

Um dos objetivos do projeto é recriar um espaço e um estilo de vida medievais na cidade de Mértola. Foram incorporados vários objetos que vão compor a simulação dos personagens virtuais (humanos e animais), num cenário dia de mercado situado dentro da muralha na cidade de Mértola. Pensando neste argumento, foram pesquisados diversos objetos da época, que pudessem ser utilizados na cidade.

4.3.1 Elementos

Foram adicionados vegetação, água no rio e a simulação de cardumes. O processo de plantação de árvores é semelhante ao processo de aplicação de texturas explicados no Anexo A.3. O Unity 3D disponibiliza vários *prefabs* para a modelação de água, todos eles incluídos no *package environment*, já referido na secção anterior. Todos os modelos funcionam como um plano (alguns quadrangulares, outros circulares) que deve ser colocado ao nível do rio desejado.

Para a simulação de cardumes foi utilizado o modelo gratuito *Sardine*[63]da *Asset Store* do Unity 3D. E foram criados 2 *scripts*: o *flock* que é responsável pelo agrupamento dos peixes, e o *script global flock*, responsável por limitar a zona por onde os peixes nadam.

Foram adicionados também, diversos elementos como frutas e comida que pudessem existir naquela época e naquele local. Atenção especial foi dada ao mercado medieval[64], situado no centro da cidade. O mercado foi pensado de maneira que existisse espaço suficiente dentro dos limites do terreno para que os personagens virtuais pudessem executar os seus movimentos

Figura 4.23: Banca medieval para venda de fruta

63 Sardine: https://assetstore.unity.com/packages/3d/characters/animals/sardine-37963

64 Elementos do mercado medieval: https://assetstore.unity.com/lists/medieval-models-20655

recriando as atividades de compra e venda num mercado da época. Partindo desse argumento foram adicionadas no total de 9 bancas no mercado (4 para venda de frutas, 2 para venda de comida, 1 para venda de peixes, 1 para venda de tecidos, 1 para armazenamento de bebida).

A Figura 4.23 mostra 2 bancas que representam a venda de frutas. Encontrou-se na *Asset Store do* Unity um pacote com 31 tipos de frutas[65], sendo todas *low poly*, ou seja, todas tem um baixo número de polígonos.

A Figura 4.24 mostra uma das bancas responsáveis pela comida, nela é possível observar um conjunto de alimentos que eram consumidos na época e na parte direita da figura temos uma banca de venda de tecidos.

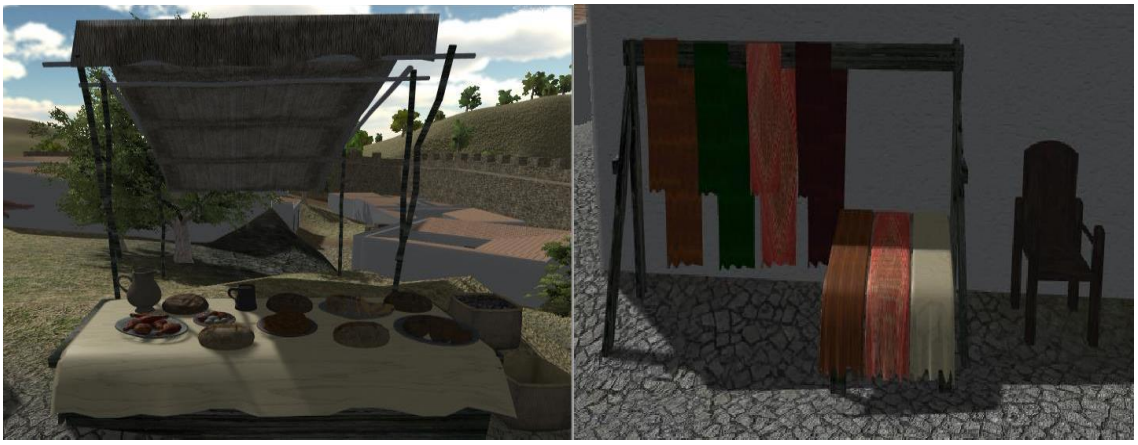


Figura 4.24: Banca medieval para venda de alimentos e a direita a barraca responsável pela venda dos tecidos medievais.

A localização exata do mercado medieval não era conhecida, portanto, foi necessário avaliar vários aspetos do terreno. Foram escolhidas zonas menos íngremes do terreno, com intuito de uma melhor integração e navegação dos personagens virtuais. A zona escolhida situa-se no centro da cidade de Mértola, limitada pela muralha. O posicionamento das bancas, também foi pensado de maneira que se evitasse um deslocamento muito grande dos avatares, com intuito de

Figura 4.25: Posição das 9 bancas no mercado medieval

65 Frutas: https://assetstore.unity.com/packages/3d/props/food/fruits-and-vegetables-pack-49035

evitar que os personagens demorassem a chegar ao seu objetivo. A Figura 4.25 mostra o posicionamento das bancas no mercado medieval.

Foi tida em consideração também a parte externa da muralha, em particular, a região que está delimitada entre a muralha e o rio Guadiana. Foi colocado um barco de pescadores no rio (Figura 4.26).

Figura 4.26: Barco de pescadores com peixes no rio

4.3.2 Personagens Autónomos

Para obter os modelos dos HV, foram utilizadas três ferramentas: Mixamo Fuse, DazStudio e 3DSMax.

O DazStudio foi utilizado para a criação de 2 personagens virtuais, uma criança e um adulto, que foram utilizados nos testes preliminares.

Figura 4.27: Avatar adulto no DazStudio preparado para ser exportado e a direita da figura um avatar criança.

Para obter os HV da versão final da ferramenta foram utilizados o 3DSMAX e o Mixamo Fuse. Este último foi escolhido porque permite criar vários personagens, com características e roupas distintas, com um esqueleto e com um controlador de animações já programadas para o esqueleto.

A criação dos personagens foi feita seguindo as etapas:

1. Foi utilizado a ferramenta Mixamo fuse para gerar o HV que de seguida se importou para o 3DSMAX no formato .OBJ.

2. No 3DSMAX adicionam-se acessórios, como por exemplo (o barrete), e altera-se a geometria das roupas (saia), por exemplo para reduzir o número de vértices, utilizando *MultiRes modifier*[66], funcionalidade do 3DSMAX responsável pela redução da sobrecarga de memória necessária para renderizar modelos. A exportação do personagem foi feita através da opção Embed media[67], que permite incluir (ou incorporar) todas as texturas no ficheiro .FBX.
3. Faz-se o *upload* do ficheiro .FBX para a ferramenta Mixamo, onde se realiza o *auto rig* do esqueleto.
4. Finalmente é feito o *download* do modelo com o *rigging* e o *skinning* feitos, e o personagem virtual está pronto para ser importado para o Unity 3D.

A Figura 4.28 resume estas etapas sequenciais.

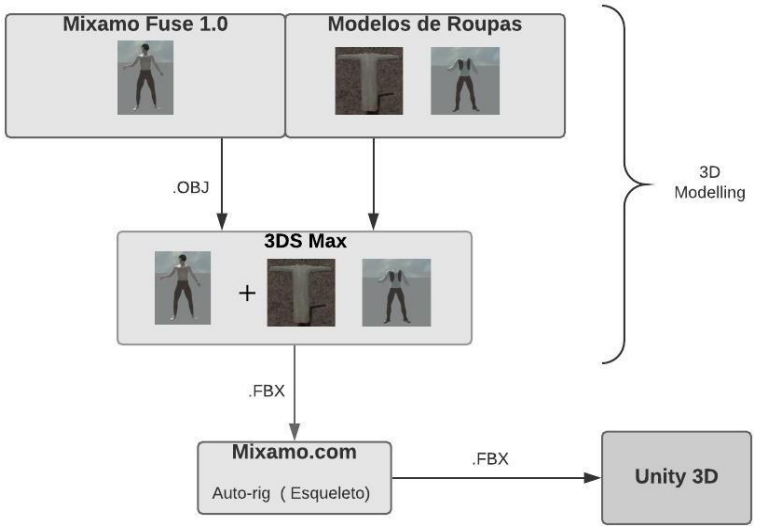


Figura 4.28: Pipeline de criação de personagens e de importação.

[66] MultiResmodifier: https://knowledge.autodesk.com/pt-br/support/3ds-max/learn-explore/caas/CloudHelp/cloudhelp/2017/PTB/3DSMax/files/GUID-DFD94E0F-7B08-4945-A176-42B49B8F458E-htm.html

[67] Embedmedia https://knowledge.autodesk.com/pt-br/support/3ds-max/troubleshooting/caas/CloudHelp/cloudhelp/2016/PTB/3DSMax/files/GUID-FB993022-FD65-4ABD-8AF3-7F2CF613D71D-htm.html

Aplicando o pipeline anterior foram criados 7 avatares (5 homens e 2 mulheres) representados na Figura 4.29.

Figura 4.29: Personagens virtuais medievais.

Como um dos objetivos do projeto é recriar o ambiente da idade média era necessário incluir modelos de animais usados nos meios de transporte da época como cavalos e burros.

Usou-se um modelo gratuito de cavalo animado (Figura 4.30) da *Asset Store* do Unity3D e um modelo de burro adquirido (Figura 4.30) no site arteria3d[68]. Ambos os modelos foram obtidos com as respetivas animações sendo posteriormente integrados na cidade virtual de Mértola.

Figura 4.30: Cavalo e Burro importados para o Unity3D.

Animação

O Unity 3D permite que algumas animações sejam realizadas diretamente no seu editor. No entanto, este editor é ideal para pequenas animações e não para animações complexas, como um movimento de um humanoide.

68 Arteria3d: https://arteria3d.myshopify.com/

Pensando nisso foram importadas da *Asset Store* do Unity as animações para o cavalo, e do Mixamo, foram importadas animações para os humanos virtuais, tais como sentar, andar, subir, correr, pular. Foi utilizada também a biblioteca da Universidade Carnegie Mellon, a moCap Library[69], para compor um conjunto de animações, visando obter variabilidade de animações.

A Figura 4.31 mostra o *Pipeline* utilizado para as animações dos personagens.

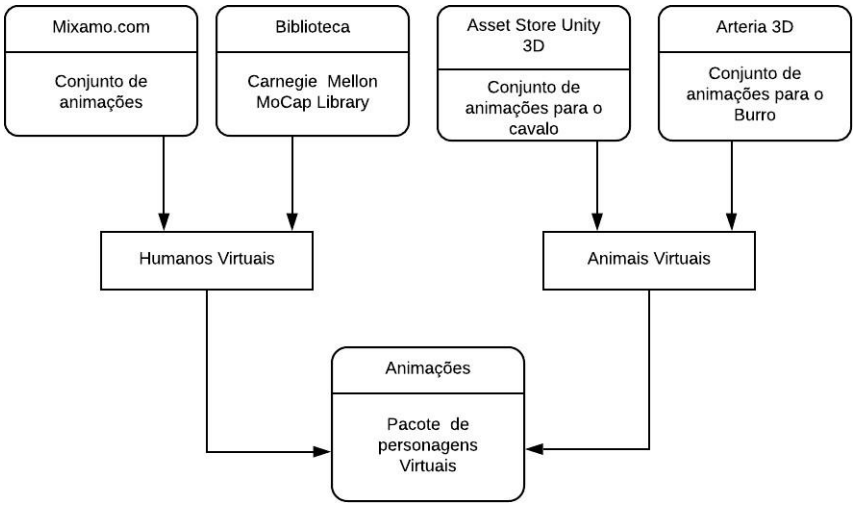


Figura 4.31: *Pipeline* utilizado para as animações dos personagens

Escolhido o personagem e a animação desejada, é possível no Mixamo configurar o movimento alternando parâmetros como a velocidade, amplitude de braços, entre outros, e descarregar o modelo para ser importado para o Unity (ver Figura 4.32). Para o projeto desenvolvido, além das animações mencionadas anteriormente, foram necessárias as seguintes animações: parado *(Neutral Idle*), recolha de um objeto (*Taking Item*), argumentação, cumprimentar (*greetings).*

No Mixamo é possível importar modelos que já incluam um esqueleto ou tirar partido do sistema automático de definição do esqueleto.

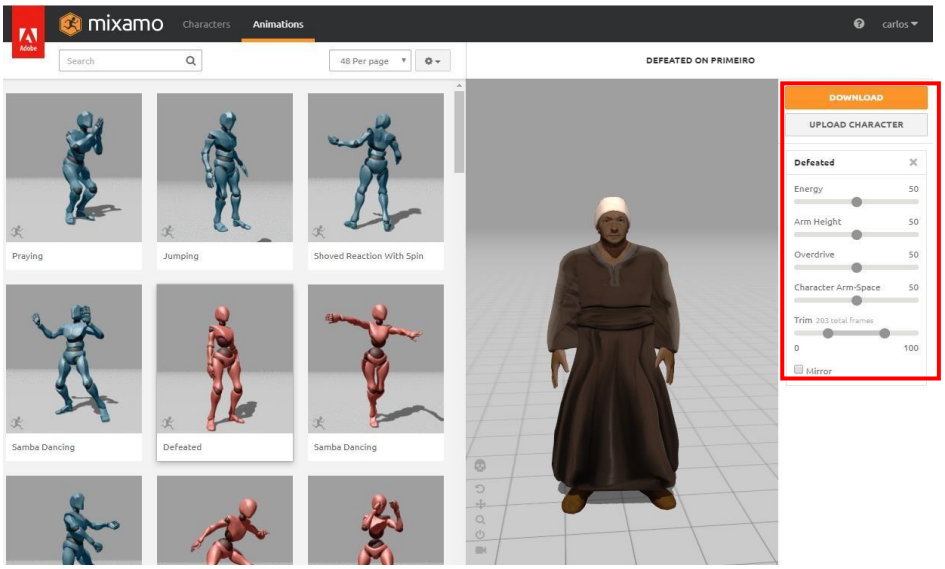


Figura 4.32: Configuração de parâmetros da animação da caminhada.

69 Mocap Library: http://mocap.cs.cmu.edu/

Uma vez ajustados os parâmetros para um movimento natural, e usar o botão *download.* Surgirá a janela que permite escolher o tipo de modelo a descarregar.

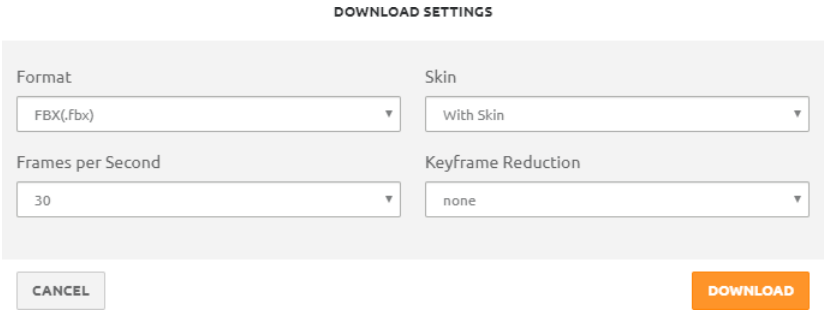


Figura 4.33: Configuração das opções do modelo a descarregar.

Para o projeto, foi escolhido o formato FBX for Unity, com a inclusão do modelo (*with skin*) de acordo com a Figura 4.33. Isto significa que o ficheiro exportado terá em consideração alguma compatibilidade com o Unity, sendo que para além das animações será exportado o modelo original depois de adicionado o esqueleto (se o avatar importado não tiver). As duas outras opções permitem aumentar o número de *frames* por segundo (para aumentar a qualidade da animação) e reduzir as *frames* principais da animação, o que pode gerar animações mais compactas, porém menos fluidas.

Controlo de animações

Cada personagem utilizado tem bastantes texturas, pelo que, para simplificar o seu uso, assim como sua posterior importação, foi criado um pacote Unity designado HumanosVirtuaisMértola.unitypackage, que as inclui juntamente com os respetivos materiais. Assim, o 1º passo para a inclusão dos personagens na cidade de Mértola é a importação do referido pacote.

Os pacotes do Unity consistem em ficheiros comprimidos que incluem os ficheiros relevantes (neste caso concreto, as imagens, e os materiais), bem como um conjunto de ficheiros de meta informação. A criação destes pacotes pode ser feita a partir do próprio Unity, usando Assets > Export Package. Por sua vez, a importação de pacotes incluídos no próprio Unity é feita a partir da lista de pacotes disponível em Assets > Import Package. Este mesmo menu inclui a opção *Custom Package*, que permite escolher um pacote disponível no sistema de ficheiros e realizar a sua importação. Depois de importado, ficará disponível no projeto uma pasta HumanosVirtuaisMértola, que contém as pastas *Materials* e *Textures.*

De seguida, serão importadas as animações. Todos os ficheiros obtidos da Mixamo (ou os ficheiros da *Mocap Library*) devem ser arrastados para a pasta HumanosVirtuaisMértola dentro do separador Projeto *(*ou usada a opção *Import New Asset* do *menu Assets* para cada um dos ficheiros). Embora habitualmente não seja importante a pasta em que são colocados os recursos importados, neste caso, as texturas foram importadas previamente e é fundamental que o Unity as reuse e não tente criar novos materiais.

Antes de usar as animações é necessário, para cada uma, configurá-la como sendo do tipo humanoide. Para isso, deverá selecionar todas as animações importadas, no separador projeto, e aceder no *Inspector* à opção *Rig* e indicar que o tipo de animação é humanoide, tal como apresentado na Figura 4.34. A cada alteração, deverá usar o botão *Apply.*

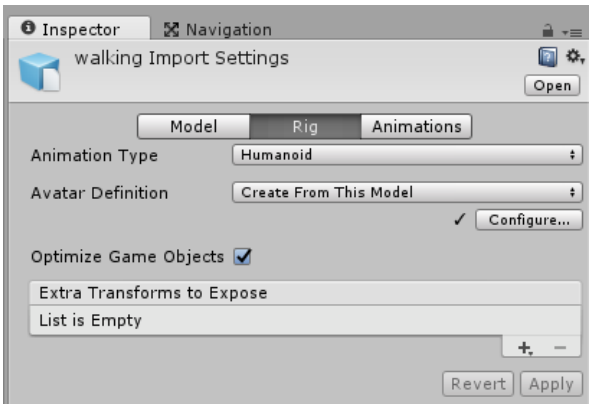


Figura 4.34: Opção para animação humanoide.

De seguida, é necessário arrastar os personagens virtuais para a cena. Para que o projeto ficasse computacionalmente mais leve e para evitar chamadas a *scripts* desnecessárias, optou-se por ter dois tipos de avatares autónomos: principais e figurantes. Estes últimos têm somente uma animação enquanto os principais podem exibir várias animações.

Como exemplo ilustrativo usamos o modelo virtual feminino designado por Safira figurante (Figura 4.35) que é um objeto composto, ou seja, não é apenas um objeto mais uma árvore de objetos, que pode ser expandida na *Hierarchy*. O objeto de topo inclui os componentes *Transform* e *Animator*. O *Animator* é responsável por associar um controlador de animações *(Animator Controller)* a um objeto.

No caso da personagem virtual Safira figurante, o seu único objetivo será percorrer o caminho entre dois pontos previamente estabelecidos, sendo necessária uma única animação. No anexo B.1 pode encontrar-se a descrição pormenorizada da sua animação.

Com objetivo de compor o cenário com animais da época adotando uma variedade de comportamentos, foram criados 2 estados chamados *DonkeyWalk* e *Graze* cada suas respetivas animações. O resultado pretendido era que o Burro caminhasse por um determinado caminho e quando chegasse ao objetivo, teria que parar e pastar. As transições vão depender do estado do personagem virtual, que será representado por um parâmetro. Pode consultar-se o Anexo B.2 para a descrição do processo.

Figura 4.35: Avatar Safira figurante na cena.

Como referido, no projeto temos 2 tipos de animais, cavalo e burro, e criaram-se variedades dos mesmos como por exemplo, o burro de carga. O mecanismo de animações do cavalo é análogo

ao explicado no Anexo B.2, a diferença encontrada são nas animações, onde o cavalo adota 2 estados chamados *Andar* e *Pastar*, cada estado com suas respetivas animações.

Navmesh da cidade de Mértola

Para movimentar os personagens virtuais (humanos e animais) no terreno, é necessário saber de que forma, estes podem descolar-se (por exemplo, não deverá poder andar sobre a água), bem como que encostas poderá subir ou descer, de acordo com a inclinação do terreno. Para além desta informação, a cada posição possível, é necessário saber qual o percurso que os personagens devem percorrer para se deslocar de um a outro ponto da cidade. O cálculo deste percurso pode ser feito diretamente usando vetores, mas não será fácil ter em conta as zonas que os personagens não podem percorrer.

Este é um problema genérico na maioria dos jogos e simulações que envolvem inteligência artificial. A maioria dos jogos utilizam algum algoritmo para calcular a melhor rota entre dois pontos. O A*[70] é um algoritmo famoso tanto na indústria como no uso académico pela sua facilidade de implementação e baixo custo de memória, porém algoritmos mais recentes podem ajudar a melhorar os resultados obtidos como por exemplo o *NavMesh*.

O Unity inclui um módulo de inteligência artificial que implementa o cálculo de caminhos sobre um conjunto de objetos. Para que este cálculo de caminhos funcione, é necessário criar uma *NavMesh*, que corresponde a uma malha, semelhante à usada para o *rendering* dos objetos, mas que, neste contexto é utilizada para o cálculo de caminhos. O *NavMesh* é uma grelha, constituída apenas por triângulos, o Unity sabe que se pode deslocar entre dois triângulos desde que ambos façam parte das zonas por onde os personagens podem caminhar e exista um conjunto de triângulos com esta mesma propriedade que liguem os dois triângulos iniciais (sendo que há caminho entre dois triângulos sempre que estes partilhem uma aresta).

O controlo da navegação sobre a *NavMesh*, é configurado no separador Navegação *(Window > Navigation)*, como explicado no (Anexo B.3).

Nessa altura, o Unity irá mostrar, na cena, as zonas por onde os personagens vão poder caminhar (Figura 4.36).

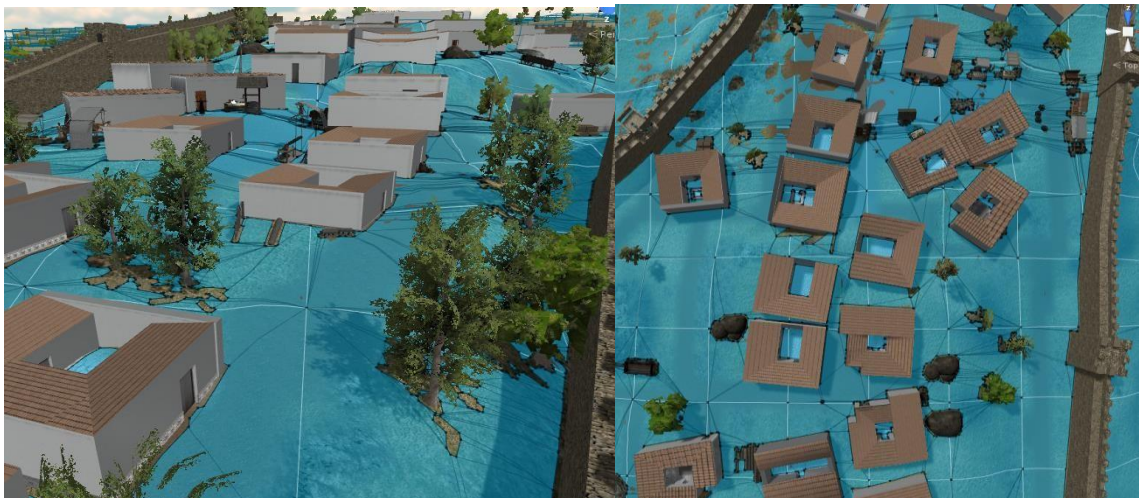


Figura 4.36: Mostra as zonas por onde os personagens vão poder caminhar.

Um personagem não jogável (do inglês, *Non-player Character* ou NPC) é um personagem dentro de um jogo ou simulação que não pode ser controlado por um jogador

[70] A*: https://www.redblobgames.com/pathfinding/a-star/introduction.html

mas que fazem parte de alguma forma do cenário. Normalmente os NPCs são dotados de alguma inteligência artificial, podendo interagir ou não com cena do jogo/simulação.

No caso de Mértola foi preparado um mecanismo de NPC, que é possível adaptar em qualquer circunstância, desde que se utilize o *NavMesh*, explicado anteriormente. Foram criados 5 *scripts*, o NPCBaseFSM, o BurroAI, os estados patrol, chase e attack, gerando a máquina de estados NPCFSM.

Este conjunto de *scripts* formam o que designamos por BurroAI (burro inteligente), e funciona da seguinte maneira: o burro inteligente tem uma zona de interesse, ou seja, são definidos pontos no mercado da cidade de Mértola onde ele irá deambular, ele percorrerá em sequência os pontos de 1 a 4. Correspondendo ao estado de patrulha do burro inteligente, quando um outro animal entra na zona de interesse previamente definida, o BurroAI começa a perseguição, e se o animal se aproximar, o BurroAI passa para o estado de ataque, este estado só terminará quando o animal, apresentar uma certa distância (condição ajustável). Porém a perseguição continuará até o animal sair do raio de ação do BurroAI.

Este mecanismo pode ser implementado também para os Humanos Virtuais, sendo necessário alterar somente as animações. A Figura 4.37 mostra a máquina de estados do BurroAI e a Tabela 9 mostra as transições entre os estados.

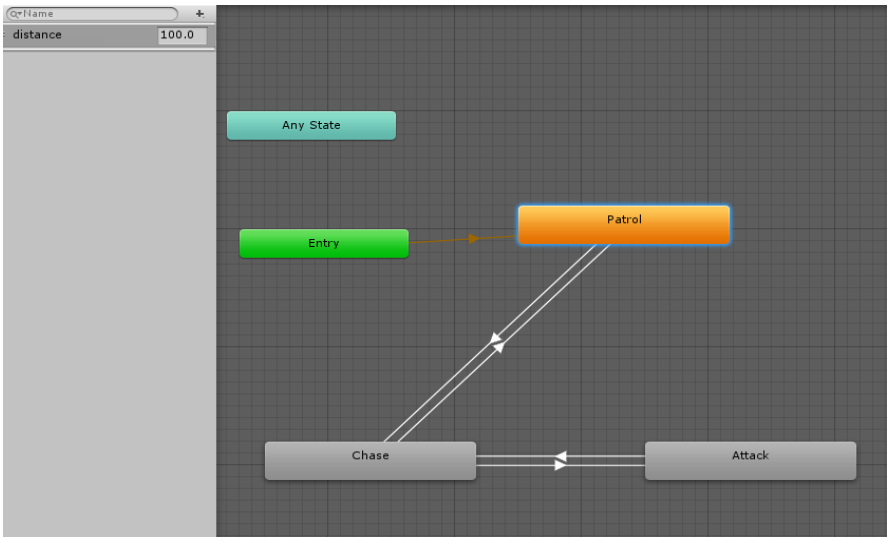


Figura 4.37: Controlador de animações do BurroAI.

Tabela 9: Transições do controlodor do BurroAI.

Origem	→	Destino	Condição
Patrol	→	*Chase*	distance < 20 (*Less*)
Chase	→	*Attack*	distance < 2 (*Less*)
Attack	→	*Chase*	distance > 2 (*Greater*)
Chase	→	*Patrol*	distance > 20 (*Greater*)

Com este mecanismo criado, o próximo passo foi criar uma solução onde este mecanismo pudesse ser testado. Para isso foram criados 2 burros, BurroAIPerseguidor e o BurroAIPerseguido.

O BurroAIPerseguidor tem a missão de andar por 4 zonas determinadas (*waypoints*), que podem ser ajustáveis de acordo com a necessidade. O comportamento inicial adotado pelo BurroAIPerseguidor, é o estado de patrulha, portanto o ataque e a perseguição ao BurroAIPerseguido somente acontecerá quando este entrar na sua zona de acção.

Por sua vez, o BurroAIPerseguido também tem a missão de percorrer as 4 zonas predeterminadas, com isso a certa altura ambos vão encontrar-se. A Figura 4.38 mostra o encontro dos dois burros, onde um persegue o outro, assim como dois pontos que foram determinados para que ambos realizem a sua missão.

Figura 4.38: Mostra o ponto de encontro dos dois burros.

O teste foi satisfatório, como mencionado anteriormente este mecanismo é completamente adaptável a diferentes tipos de personagens virtuais, (humanos e não humanos). Porém, existe uma particularidade, ele só funciona através da dependência dos personagens, é preciso referenciar "quem" vou buscar diretamente, ou seja, quando a simulação é iniciada, o BurroAIPerseguidor já sabe "quem" tem que procurar, para isso é preciso que os 2 personagens virtuais (BurroAIPerseguido e BurroAIPerseguidor) estejam na cena que será iniciada.

No caso do projeto temos uma particularidade, temos burros que não vão iniciar a cena, ou seja teremos um burro a andar pelos pontos e outro que somente entrará na cena depois, se o utilizador tiver escolhido. Para verificar se a perseguição é realizada, foram feitos os seguintes testes:

1. **Cenário com 2 burros na mesma cena:** O burro1 começa na cena e tem o objetivo de procurar o burro2.
2. **Cenário com 2 burros um na cena e outro não:** O burro1 começa na cena e tem o objetivo de procurar o burro2 que não está na cena.
3. **Cenário com 2 burros que não estão na cena:** O burro1 tem o objetivo de procurar o burro2, porém não estão na cena.

Dos 3 testes anteriores, somente o 1 foi realizado com sucesso. No teste número 2, o burro1 começa na cena com o objetivo de buscar o burro2 que entra na cena posteriormente, apesar do burro1 ter como referência o burro2, ele não sai a procura. E o mesmo acontece com teste 3, onde temos 2 Burros com objetivos já definidos, e quando entram na cena, também não cumprem. Isto aconteceu porque é a maneira que o Unity trata a dependência de objetos (ambos precisam estar na mesma cena). Para resolver esse problema, foi concluído que era necessário remover as dependências entre os objetos, depois de uma profunda pesquisa, foi encontrada uma solução, a utilização de *scriptable objects*[71] , que é um tipo de componente (*script*) que pode conter dados e métodos, como um componente normal, mas não precisa existir na cena para poder executar seus métodos ou ter acesso aos dados de outros componentes da cena.

Pensando na solução deste problema foi utilizado o *scriptable objects as variables*, sendo um dos maiores benefícios o de não ser necessário obter referências dos objetos afetados por essa variável, ou seja, consigo conectar um objeto com outro sem se referenciar.

Foi então criado uma classe que deve herdar do *scriptable objects*. De acordo com a Figura 4.39.

```
1
2 using UnityEngine;
3
4 [CreateAssetMenu(fileName = "NewTransformVariable", menuName = "Variables/Transform", order = 6)]
5
6 public class TransformVariable : ScriptableObject {
7     // crio uma variavel onde, quem a tiver referenciada, buscará quem tiver o script ObjPerseguido tendo como target o Alvoperseguido
8     [HideInInspector]
9     public Transform Value;
10 }
11
```

Figura 4.39: Scriptable object.

O processo para dar comportamento autónomo aos personagens está descrito no Anexo B.4.

Personagens

Antes da criação da interface da aplicação Mértola Virtual testou-se o ambiente virtual gerado com uma simulação de uma pequena amostra de avatares cujo comportamento é baseado na execução de *scripts* de Inteligência Artificial.

A simulação baseia-se num dia de mercado na cidade de Mértola. Para isso foram criados vários tipos de personagens virtuais (figurantes, compradores e vendedores, animais). A seguir vai ser explicado cada um dos tipos dos personagens.

Personagens figurantes:

Para que o projeto ficasse computacionalmente mais leve e evitar chamadas a *scripts* desnecessários, foi estabelecido que todos os todos os humanos figurantes do cenário, teriam somente uma animação, eventualmente é possível ter outras animações, bastando acrescentá-las na máquina de estados de cada avatar.

Foram criados 6 personagens figurantes, 2 personagens femininos e 4 masculinos (Safira figurante, Yasmine figurante, Ibraim figurante, Mohamed figurante, Mustafa figurante, Kaled figurante).

[71] scriptable objects: https://docs.unity3d.com/ScriptReference/ScriptableObject.html

No caso da personagem virtual Safira figurante, o seu único objetivo na cidade de Mértola será percorrer o percurso entre dois pontos previamente estabelecidos. Já a personagem Yasmine figurante, estará a observar as pessoas que estão no mercado.

O personagem Ibraim figurante tem o objetivo de sentar-se e observar o mercado olhando para direita e esquerda, o Mohamed figurante, que se encontra logo atrás do personagem Ibraim figurante, executando as animações de se baixar e recolher frutas no chão. A Figura 4.40 mostra os 2 personagens virtuais a executar as suas respetivas animações.

Figura 4.40: Personagens Virtuais em ação.

Os personagens virtuais Mustafa e Kaled figurante, ambos estão sentados, o Mustafa está sentado ao lado da banca de roupas no mercado da cidade de Mértola, já o Kaled figurante está sentado em frente a uma casa em outra parte da cidade.

Personagens Animais:

Foram criados 5 personagens animais, (BurroFigurante, BurroCarga, BurroAIPerseguidor, BurroAIPerseguido, cavalo).

O BurroFigurante tem como objetivo andar por um determinado caminho e quando chegar ao destino, terá que parar e pastar. Já o BurroCarga tem o objetivo de andar entre dois pontos podendo ser estender a mais pontos, para isso foi criado o *script Waypoints*, na qual é possível escolher os lugares onde o BurroCarga irá passar, este mesmo *script* poderá ser utilizado no cavalo como mencionado anteriormente, uma alternativa pensada, foi ter o BurroCarga a seguir um personagem humano para que isso acontecesse foi necessário o *script finder*, explicado no Anexo B.4.

Já os personagens BurroAIPerseguidor, BurroAIPerseguido e cavalo foram explicados anteriormente. A Figura 4.41 mostra o BurroFigurante e o BurroCarga a realizar seus respetivos objetivos.

Figura 4.41: BurroFigurante a chegar no seu objetivo e o BurroCarga próximo do objetivo *waypoint* 3

Personagens compradores

Foram criados 2 personagens os compradores, 1 personagem masculino e 1 feminino (Ali, Yasmine). Para ambos personagens foram criados 3 tipos de animações (andar, suspirar aliviado e gesticular), com essas animações foi gerada a máquina de estados chamada "andar".

Para que os personagens possam encontrar o ponto de destino foi criado o *script PathFinder*, uma versão análoga ao *script finder*, e para controlar o comportamento dos personagens compradores foi criado o *script CompradorBehaviour* para que este script funcione o avatar terá que ter o script *PathFinder* adicionado a ele.

Quando a simulação é iniciada, os compradores identificam todos os vendedores em cena e visita a todos na ordem do mais próximo ao mais distante e ao colidir com um vendedor, executa uma animação que simula uma conversa utilizando a máquina de estados "andar" (Figura 4.42).

Figura 4.42: Personagens virtuais comprador e vendedor.

Personagens vendedores

Foram criados 4 personagens do tipo vendedores (Omar, Kaled, Youssef, Ibraim). A máquina de estados "vendedor" foi criada para os personagens vendedores e possui 4 animações diferentes (reação, argumentar, suspirar aliviado e gesticular).

Para que o vendedor possa executar o seu objetivo foi preciso criar um *script VendedorBehaviour,* onde o vendedor dispara uma animação quando é identificado a colisão com um comprador. Ao executar a animação, o vendedor vira para o último comprador que se aproximar.

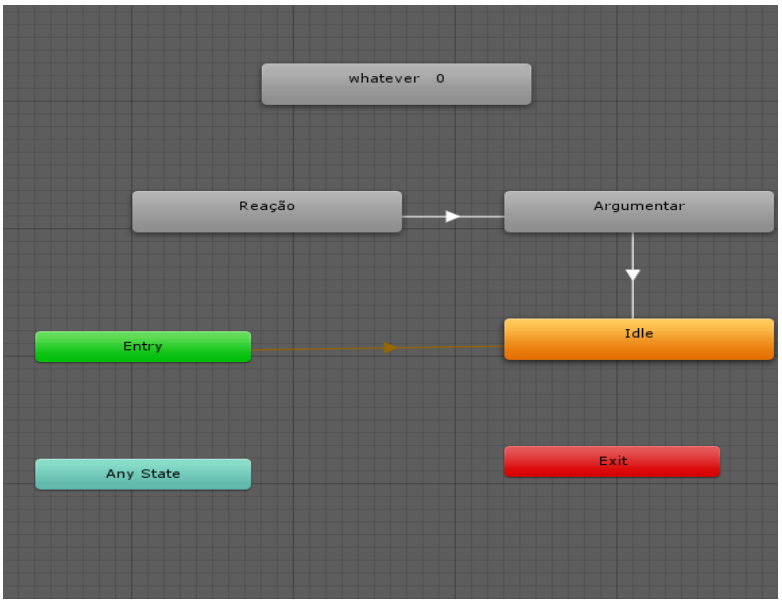


Figura 4.44: Máquina de estados vendedor.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class VendedorBehaviour : MonoBehaviour
{

    private Animator anim;

    private void Start()
    {
        anim = GetComponent<Animator>();
    }

    public void StartCompradorAnim()
    {
        anim.Play("Reação");
    }

    public IEnumerator TurnToComprador(Transform target)
    {
//        transform.LookAt(target);
        float initTime = 0;
        var targetRotation = Quaternion.LookRotation(new Vector3(target.transform.position.x, transform.position.y, target.transform.position.z)/
        while (true)
        {
            transform.rotation = Quaternion.Slerp(transform.rotation, targetRotation, 1.3f * Time.deltaTime);
            if (initTime >= 3f)
            {
                yield break;
            }
            yield return new WaitForEndOfFrame();
        }
    }
}
```

Figura 4.43: *Script VendedorBehaviour*

Para que o vendedor possa executar o seu objetivo foi preciso criar um *script VendedorBehaviour*, onde a animação do vendedor é lançada quando é identificado a colisão com um comprador. Ao executar a animação, o vendedor vira para o último comprador a se aproximar.

Para evitar que a simulação se repita, ou seja, ainda que se escolha o mesmo número de personagens e animais exatamente da mesma forma, não teremos 2 simulações iguais, o local de nascimento (*Spawn)* de todos os personagens é calculado de forma aleatória.

4.4 Interface

Quando um jogo ou simulação é iniciado, além de um ou mais ecrãs com informação sobre os seus produtores e nalguns casos, uma imagem com o título, é apresentado aos utilizadores um menu com algumas opções, como a possibilidade de jogar num ou diferentes modos de jogo, a configuração da qualidade gráfica, etc., e a possibilidade de cancelar o jogo ou simulação e voltar ao sistema operativo.

A forma mais simples de gerir diferentes zonas de uma simulação ou jogo, ou diferentes ecrãs, é a criação de várias cenas. Nesta secção é ilustrada a criação de uma cena para apresentar aos utilizadores a interface da aplicação, que foi cuidadosamente pensada para ser utilizada em diferentes projetos dentro do âmbito do Unity3D, oferecendo oportunidades de aprendizagem no estudo histórico e conhecimento de ambientes antigos e proporcionando ao utilizador uma sensação de presença numa simulação de uma cultura que recria cenários e comportamentos da época. A Figura 4.45 mostra o menu que foi construído.

O sistema de avatar criado permite que os visitantes entrem no mundo selecionando uma classe de (compradores e vendedores) assim como seu género, é possível escolher no máximo 10 avatares do tipo compradores e 6 avatares do tipo vendedores.

O Menu permite ainda que o utilizador escolha o idioma, podendo ser entre Português e o Inglês. Através das teclas F1 e F2 o utilizador pode escolher a navegação em 1ª pessoa e 3ª pessoa respetivamente. Já as teclas F3 e F4, permitem visualizar a simulação em diferentes pontos de vista.

Figura 4.45: Menu inicial da simulação.

Para que o menu principal da aplicação fosse gerado foi necessário criar 3 *scripts* (*MainMenu*, *Container* e o *ExitMenu).*

O *script MainMenu* é responsável pela inserção dos referidos personagens assim como a validação dos mesmos, verificando se todos os campos foram preenchidos, caso contrário que será preenchido 1 por omissão (1 vendedor, 1 comprador, 1 animal e local escolhido é o mercado). Após essa validação (Figura 4.46) chama o *script Container* que guarda as informações preenchidas no menu para serem carregadas na cena da simulação, assim, instancia os objetos (compradores, vendedores, animais) nos devidos lugares de nascimentos que são calculados de forma aleatória.

Já o *script ExitMenu é* responsável pelo fim da simulação, assim como o seu recomeço.

```
public void Iniciar()
{
    //verifica se cada campo esta sendo preenchido senão for, ele coloca 1 de cada tipo por defeito
    if (personagens_vendedores.Count == 0)
    {
        personagens_vendedores.Add(new Personagem("Vendedor1", ReturnGameObject("Vendedor1")));
    }
    if (personagens_compradores.Count == 0)
    {
        personagens_compradores.Add(new Personagem("Comprador1", ReturnGameObject("Comprador1")));
    }
    if (personagens_animais.Count == 0)
    {
        personagens_animais.Add(new Personagem("Animal2", ReturnGameObject("Animal2")));
    }
    if (personagens_ambiente == "")
    {
        personagens_ambiente = "Mercado";
    }
    //        if (personagens_vendedores.Count > 0 && personagens_compradores.Count > 0 && personagens_animais.Count > 0 && personagens_ambiente != "") {
    GameObject.Find("Container").GetComponent<Container>().Run(personagens_compradores, personagens_vendedores, personagens_animais, personagens_ambiente);
    //        } else
    //        {
    //            FeedbackError("Deve adicionar ao menos um item em cada objeto");
    //        }
}
```

Figura 4.46: Função que chama o *script container*

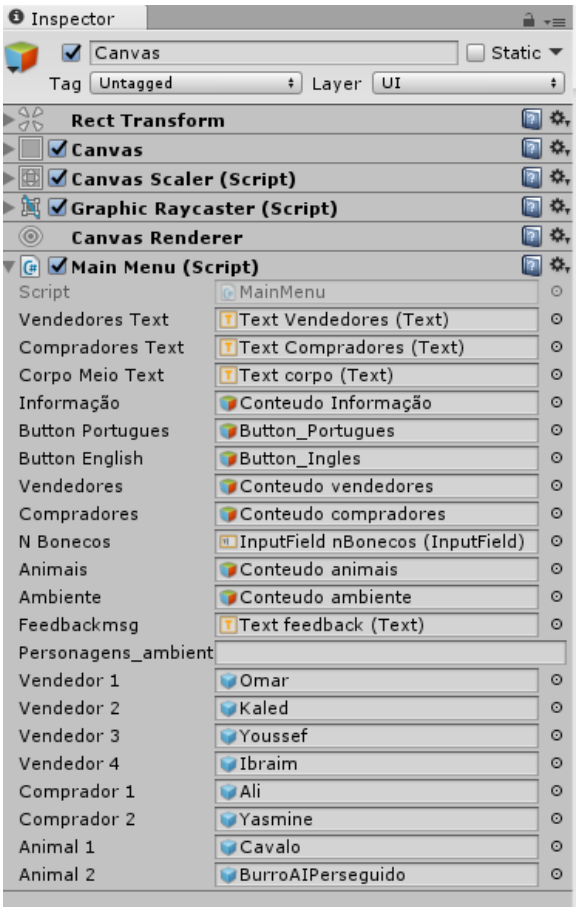


Figura 4.47: *Inspector* do *MainMenu*

Como é possível observar na Figura 4.47, quando o *script MainMenu* é incorporado, vem vazio, sendo necessário preencher os respetivos vendedores, compradores e animais, assim como os botões que compõe o menu, com isso, ele pode ser adaptado a qualquer outra situação. A secção a seguir mostra as funcionalidades da Interface.

4.4.1 **Funcionalidades da Interface**

Após entrar na aplicação, são apresentadas ao utilizador várias opções. Estas opções variam de acordo com a escolha do utilizador (Figura 4.48).

Figura 4.48: Escolha de personagens

Como referido anteriormente, utilizador tem a opção de escolher os idiomas português ou inglês, uma vez escolhido entra na aplicação e o próximo passo será escolher os números de personagens que irão compor a simulação, no caso mostrado na Figura 4.48, o utilizador poderá escolher até no máximo 6 vendedores, podendo ser 6 de um tipo.

Uma vez escolhido o número de vendedores e compradores, o próximo passo será a escolha dos animais que pode ir podem ser até num máximo de 10 (Figura 4.49).

Figura 4.49: Escolha dos animais.

Depois de escolhido o número de personagens e animais, o próximo passo será escolher onde a simulação irá começar podendo ser no “mercado” ou na “rua”. A escolha da local, representa o ponto de início da simulação, ou seja, onde a câmara principal irá começar.

Figura 4.50: Escolha do local de início.

Uma vez escolhido o local de início da simulação o próximo passo será clicar no botão iniciar, o que fará executar o *script Container* que guardará as listas correspondentes, e inicia a simulação.

Figura 4.51: Simulação iniciada.

Foi criado o *script ExitMenu* que permite terminar e recomeçar a execução da simulação. Sendo associado ao *canvas* criado para este propósito, um detalhe importante, o *canvas* tem de estar desativado para que não seja visível em tempo de execução, somente quando o utilizado pressionar o botão "esc", aparecerá no ecrã o *canvas* criado com as funcionalidades desejadas, *MenuSair* gerado terá de ser colocado na cena principal e não na cena inicial que contém o menu principal.

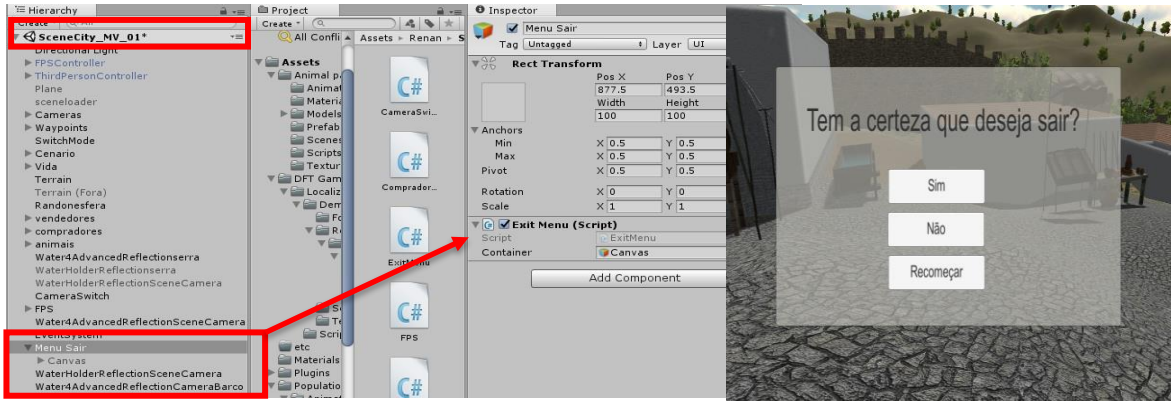


Figura 4.52: *MenuSair* criado.

Como mencionado anteriormente, a interface da aplicação permite que o utilizador escolha o idioma, podendo ser entre Português e o Inglês. Para que este mecanismo fosse implementado, existiam 2 soluções:

1. A criação de um segundo Menu, no qual teríamos que traduzir tudo, o que implicaria uma duplicação de código. Nesta opção bastaria termos um botão que ativasse o menu com o idioma preferido, desativando o outro. Esta opção foi rapidamente descartada, pelo fato de termos que duplicar código.
2. A segunda opção encontrada, foi utilizada pela DFTGames[72], disponível no AssetStore do Unity. Esta solução utiliza um dicionário (key = *value*), onde é preciso criar um ficheiro .txt com a tradução correspondente utilizando o formato mencionado anteriormente (key = *value*).

   A Figura 4.53 mostra os 2 ficheiros .txt que foram utilizados na aplicação.

[72] DFTGames: https://dftgames.com/

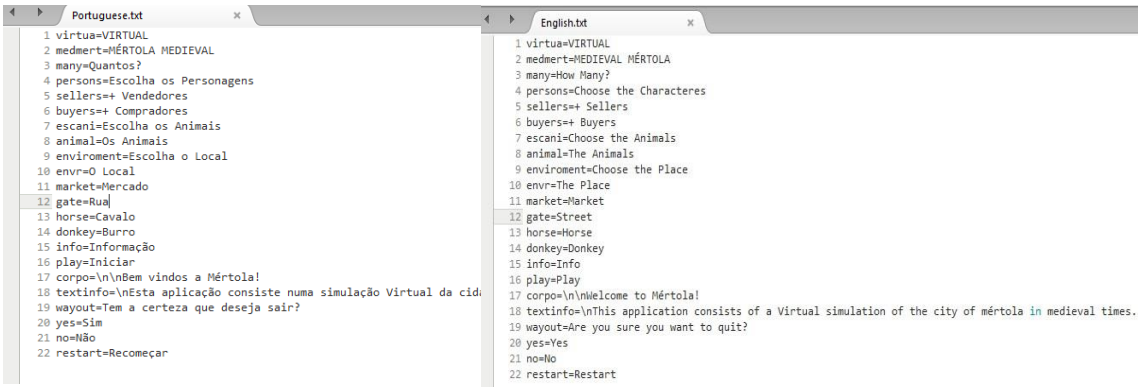
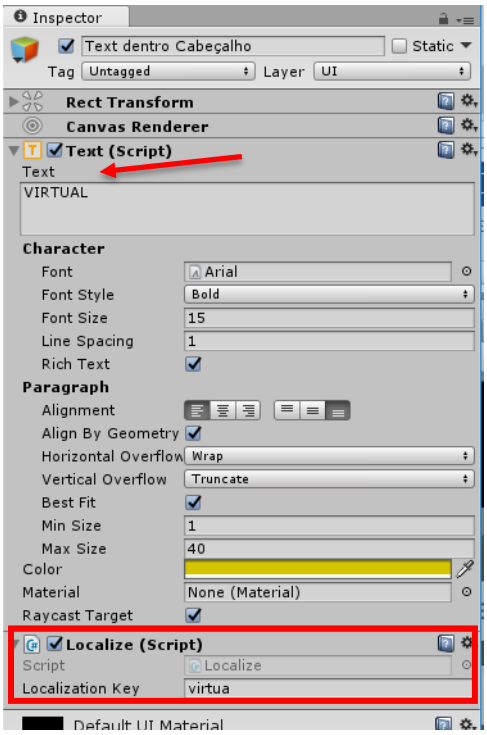


Figura 4.53: Ficheiros .txts utilizados para a tradução




```
public void activeVendedores()
{
 // verifica se o texto do componente é igual a palavra que vai ser gerada de acordo com a lingua, se for sellers entra no primeiro ciclo, cc a palavra será + Vendedore
    if (GameObject.Find ("Text Vendedores").GetComponent<Text> ().text == "+ Sellers"||GameObject.Find ("Text Vendedores").GetComponent<Text> ().text == "- Sellers")
    {
            vendedoresText.text = "- Sellers";
            buttonEnglish.gameObject.SetActive (false);
            buttonPortugues.gameObject.SetActive (false);
            corpoMeioText.gameObject.SetActive (false);
            informação.gameObject.SetActive (false);
            compradores.gameObject.SetActive (false);
            vendedores.gameObject.SetActive (true);
```

Figura 4.54: *Script* responsável pela tradução, e pedaço de código correspondente a tradução dos vendedores e compradores.

A tradução é validada, quando o utilizador seleciona o idioma desejado, para cada componente *Text* é verificado através do *localization key* a sua tradução correspondente no ficheiro .txt correspondente de acordo com a escolha inicial do idioma. Porém existe uma particularidade, o menu é composto de submenus, quando é escolhido o idioma desejado, ele realiza a tradução de todos os botões do menu principal, porém para a funcionalidade dos "vendedores e compradores" é preciso fazer uma pequena validação extra da componente *Text*, ou seja, verifica se o texto do componente é igual a palavra que vai ser gerada de acordo com o idioma desejado, e com isso ativa o menu de acordo com essa verificação. A Figura 4.54

mostra um pequeno pedaço de código responsável por essa verificação, que no caso seria se o idioma escolhido for o inglês significa que *Text* aparecerá + *sellers,* portanto ativa o submenu inglês desativando o submenu em português, caso contrário, a palavra será + vendedores, neste caso será ativado o submenu em português.

4.5 Conclusão

Neste capítulo apresentou-se um *pipeline* que identifica os passos para a conversão adequada do terreno disponibilizado no projeto anterior e os passos para a sua ampliação no Unity 3D; descreveu-se o processo de inserção de personagens autónomos no cenário explicando os *scripts* implementados para este efeito; finalmente apresentou-se a interface e descreveram-se as suas funcionalidades.

Não houve oportunidade de realizar testes com utilizadores reais, embora ao longo de todo o trabalho se tenham procedido a experimentações por parte de elementos da equipa do laboratório de investigação. Estas experiências foram determinantes para identificar melhorias necessárias.

No próximo capítulo é feita a conclusão e são apresentadas ideias para trabalho futuro.

Capítulo 5

Conclusões e trabalho futuro

A utilização de ambientes virtuais não se restringe apena as áreas lúdicas como os videojogos e os filmes. Em RV aparece cada vez aplicada em contextos culturais, em particular de recriação de civilizações antigas, como pudemos constatar na pesquisa bibliográfica descrita no capítulo 2

Neste projeto foi criada uma aplicação que visa alcançar o utilizador não especializado oferecendo oportunidades de aprendizagem no estudo histórico e conhecimento de ambientes antigos com personagens autónomos que exibem costumes e comportamentos da época. Com este sistema pretende-se oferecer ao utilizador uma experiência de fácil usabilidade que seja interessante e apelativa, mas que tem toda uma outra componente focada na utilidade, nomeadamente na divulgação da herança cultural digital.

Para a realização da ferramenta Mértola Medieval, recorreu-se a ferramentas gratuitas ou com licenças académicas de modo a minimizar os custos envolvidos. Ao longo de todo o processo houve a preocupação constante de encontrar um equilíbrio entre a qualidade dos modelos 3D e o número de polígonos, de modo a oferecer ao utilizador uma experiência credível, mas que suportasse a sua interação em tempo real.

Foram satisfeitos os seguintes objetivos:

- A introdução de processos que garantam variabilidade de comportamentos dos humanos virtuais que se movimentam em cenários tridimensionais onde foi concebida uma ferramenta útil no contexto da herança cultural digital (recriação de civilizações antigas).
- O projeto não apenas recria as casas e a muralha, mas também inclui um completo sistema de avatar característicos da época, personagens não-jogadores com inteligência artificial, vegetação apropriada e água.
- A interface foi pensada de maneira agnóstica, isto é, que se adaptam a todo o tipo de plataforma, desde que sejam no âmbito da ferramenta Unity3D, dando ao utilizador o poder de decisão e flexibilidade na escolha dos personagens, oferece também oportunidades de aprendizagem.
- Permissão para que os utilizadores mudem de participantes para espectadores de forma a mergulhar em todos os aspetos na simulação do mercado.
- Tendo em conta a preocupação de obter uma solução de baixo custo é essencial recorrer sempre que possível a ferramentas gratuitas ou com licenças académicas e que permitam fluxos de dados entre si, para isso é necessário identificar estas ferramentas, testando a integração de modelos tridimensionais criados nas distintas ferramentas.

5.1 Trabalho futuro

Como trabalho futuro, será necessário testar a aplicação com utilizadores reais de diversas idades e formações, para medir o seu potencial para a divulgação de hábitos em civilizações antigas e identificar eventuais aspetos a melhorar. Será importante realizar estes testes no ecrã de computador e num equipamento imersivo de RV.

Relativamente a novas funcionalidades a implementar:

- Incluir na interface um *minimap* interativo que permita definir facilmente uma nova posição de observação.
- Aumentar na interface o tipo de personagens disponíveis e a sua variabilidade de comportamentos (exemplos: pescadores, crianças, rebanhos de ovelhas, entre outros).
- Implementar *scripts* que simulem comportamentos credíveis por parte destes novos personagens.
- Aumentar na aplicação as simulações disponíveis (exemplos: atividade da pesca, transporte de mercadorias e bens, atividade da pastorícia, jogos e brincadeiras).
- Adaptar a interface a possibilidade de alargar ou restringir zonas de influências do comportamento dos personagens autónomos (como por exemplo: aumentar a zona em que o BurroAI vai reagir quando outro personagem virtual se aproximar).

Bibliografia

[1] Krueger, Myron W. Artificial Reality II. Reading, MA: Addison-Wesley.

[2] Refsland, S. T., Ojika, T., Addison, A. C., & Stone, R. (2000). Virtual heritage: breathing new life into our ancient past. *IEEE MultiMedia*, *7*(2), 20-21.

[3] Carvalho, A. A. D. (2016). Reconstrução digital de espaços históricos: o caso de estudo de Mértola virtual. (Master's thesis dissertation). Faculdade de Ciências da Universidade de Lisboa.

[4] Halarnkar, P., Shah, S., Shah, H., Shah, H., & Shah, A. (2012). A review on virtual reality. *International Journal of Computer Science Issues (IJCSI)*, *9*(6), 325.

[5] Zhou, N. N., & Deng, Y. L. (2009). Virtual reality: A state-of-the-art survey. *International Journal of Automation and Computing*, *6*(4), 319-325.

[6] Tori, R., Kirner, C., & Siscoutto, R. A. (2006). *Fundamentos e tecnologia de realidade virtual e aumentada* (pp. 2-82). Editora SBC.

[7] Burdea, G. C., & Coiffet, P. (2003). *Virtual reality technology*. John Wiley & Sons, Inc., New York, NY, USA, 2 edition.

[8] Morie, J. F. (2006, January). Virtual reality, immersion, and the unforgettable experience. In *Stereoscopic Displays and Virtual Reality Systems XIII* (Vol. 6055, p. 60551X). International Society for Optics and Photonics.

[9] Mazuryk, T., & Gervautz, M. (1996). Virtual reality-history, applications, technology and future.

[10] De Faria, J., Figueiredo, E., & Teixeira, M. (2014). Histórico da realidade virtual e seu uso em medicina. *Revista De Medicina*, *93*(3), 106-114.

[11] Della Croche, L., dos Santos, L. H., da Silva, D. R., Salerno, M. H., de Lima, J. F., da Silva, D. R., & Schimiguel, J. (2016). REALIDADE VIRTUAL–A VIABILIDADE DA IMERSÃO TOTAL NA ATUALIDADE.

[12] Lima, P. R. P. (2018). Jornalismo e realidade virtual: análise da série The Daily 360 do The New York Times. (Master's thesis dissertation). Faculdade de Ciências Sociais e Humanas da Universidade Nova de Lisboa.

[13] Alqahtani, A. S., Daghestani, L. F., & Ibrahim, L. F. (2017). Environments and System Types of Virtual Reality Technology in STEM: a Survey. International Journal of Advanced Computer Science and Applications (IJACSA), 8(6).

[14] Rodrigues, G. P., & de Magalhães Porto, C. (2013). Realidade virtual: conceitos, evolução, dispositivos e aplicações. Interfaces Científicas-Educação, 1(3), 97-109.

[15] Toshniwal, R., & Dastidar, K. G. (2014). Virtual Reality: The Future Interface of Technology.

[16] Martín-Gutiérrez, J., Mora, C. E., Añorbe-Díaz, B., & González-Marrero, A. (2017). Virtual technologies trends in education. *EURASIA Journal of Mathematics Science and Technology Education*, *13*(2), 469-486.

[17] Pair, J., Allen, B., Dautricourt, M., Treskunov, A., Liewer, M., Graap, K., & Reger, G. (2006, March). A virtual reality exposure therapy application for Iraq war post traumatic stress disorder. In *Virtual Reality Conference, 2006* (pp. 67-72). IEEE.

[18] Boeing: Services - Simulator Services. (2018). Obtido de https://www.boeing.com/commercial/services/flight-operations-solutions/simulator-services/#/simulator-management

[19] Sanders, D. H. (2000). Archaeological publications using virtual reality: case studies and caveats.
[20] Computer Grafica | CAPWARE. (2018). Obtido de http://www.capware.it/service/computer-grafica/
[21] Torres, J. A. R., Maciel, A., & Nedel, L. P. (2002). Uma arquitetura para animação de humanos virtuais com raciocínio cognitivo. In *Symposium on Virtual Reality. Fortaleza, Brasil: SBC.*
[22] Thalmann, N. M., & Thalmann, D. (2012, May). Virtual humans: back to the future. In *Proceedings of Graphics Interface 2012* (pp. 1-8). Canadian Information Processing Society.
[23] Portalés, C., Rodrigues, J., Rodrigues Gonçalves, A., Alba, E., & Sebastián, J. (2018). Digital Cultural Heritage.
[24] Wingard, A. (2016). An exploration of item relatedness in a user--curated corpus of digital cultural heritage.
[25] Tan, B. K., & Rahaman, H. (2009, June). Virtual heritage: Reality and criticism. *In CAAD Futures* (pp. 143-156).
[26] Sanchotene, I. S. (2007). *Técnicas de virtual heritage (VH) e as legislações brasileiras aplicadas ao patrimônio cultural-Estudo de Caso: Campo de SantAnna* (Master's thesis dissertation). COPPE, UFRJ. Rio de Janeiro.
27] Bernardes, P. (2002). *Arqueologia urbana e ambientes virtuais: um sistema para Bracara Augusta* (Master's thesis dissertation). Instituto de Ciências Sociais da Universidade do Minho
[28] Codina, F., de Prado, G., Ruiz, I., & Sierra, A. (2017). The Iberian town of Ullastret (Catalonia). An Iron Age urban agglomeration reconstructed virtually. *Archeologia e Calcolatori*, *28*(2), 311-320.
[29] Ullastret, 250 B.C. A virtual reconstruction of an Iron Age Town | ViMM. (2017). Obtido de https://www.vi-mm.eu/2017/03/13/ullastret-250-b-c-a-virtual-reconstruction-of-an-iron-age-town/
[30] City of Cagliari Historical Reconstruction. (2008). Obtido de http://www.sjmtech.net/portfolio/cagliari_storica/
[31] Carthago Nova 'El esplendor de una era' - Region de Murcia Digital. (2011). Obtido de http://www.regmurcia.com/servlet/s.Sl?sit=c,373,m,2916&r=ReP-21627-DETALLE_REPORTAJESPADRE
[32] Gonçalves, A. J. M., & Mendes, A. J. (2003). Realidade virtual na reconstrução de ambientes históricos: o Fórum Flaviano de Conimbriga. In *3rd International Conference of Information and Communication Technologies in Education*.
[33] Museu virtual leva a arte islâmica a turistas e estudantes - Ciência - Estadão. (2009). Obtido de https://ciencia.estadao.com.br/noticias/geral,museu-virtual-leva-a-arte-islamica-a-turistas-e-estudantes,355848
[34] Troiano, D., Morro, A. G., Merlo, A., & Vidal, E. V. (2014, November). From a Model of a City to an Urban Information System: The SIUR 3D of the Castle of Pietrabuona. In *Digital Heritage: Progress in Cultural Heritage. Documentation, Preservation, and Protection5th International Conference, EuroMed 2014, Limassol, Cyprus, November 3-8, 2014, Proceedings* (Vol. 8740, p. 121). Springer.
[35] Foni, A., Papagiannakis, G., & Magnenat-Thalmann, N. (2002, October). Virtual Hagia Sophia: Restitution, visualization and virtual life simulation. In *Proc. UNESCO World Heritage Congress* (Vol. 2).
[36] Kennedy, S., Fawcett, R., Miller, A., Dow, L., Sweetman, R., Field, A., ... & Allison, C. (2013, October). Exploring canons & cathedrals with open virtual worlds: The recreation of st

andrews cathedral, st andrews day, 1318. In *2013 Digital Heritage International Congress (DigitalHeritage)* (Vol. 2, pp. 273-280). IEEE.
[37] Frischer, B. (2008). The Rome Reborn Project. How Technology is helping us to study history. *OpEd, November, 10*, 2008.
[38] Sanchez, S., Luga, H., Duthen, Y., & Balet, O. (2004, January). VIBES: bringing autonomy to virtual characters. In *International Symposium and School on Advancex Distributed Systems* (pp. 19-30). Springer, Berlin, Heidelberg.
[39] Open Knowledge and the Public Interest. (2009). Obtido de https://okapi.wordpress.com/
[40] Morgan, C. L. (2009). (Re) Building Çatalhöyük: changing virtual reality in archaeology. *Archaeologies*, *5*(3), 468.
[41] Frischer, B., & Fillwalk, J. (2012, September). The Digital Hadrian's Villa Project: Using virtual worlds to control suspected solar alignments. In *Virtual Systems and Multimedia (VSMM), 2012 18th International Conference on* (pp. 49-55). IEEE.
[42] Maïm, J., Haegler, S., Yersin, B., Mueller, P., Thalmann, D., & Van Gool, L. (2007). Populating ancient pompeii with crowds of virtual romans. In *Proceedings of the 8th International Symposium on Virtual Reality, Archeology and Cultural Heritage-VAST* (No. VRLAB-CONF-2008-151).
[43] Muller, P., Vereenooghe, T., Ulmer, A., & Van Gool, L. (2005, November). Automatic reconstruction of Roman housing architecture. In *International Workshop on Recording, Modeling and Visualization of Cultural Heritage* (pp. 287-297).
[44] Kim, L. C., Lam, T. K., & Chee, C. Y. (2016). A Multi-Modal Virtual Walkthrough of the Virtual Past and Present Based on Panoramic View, Crowd Simulation and Acoustic Heritage on Mobile Platform. *International Journal of Computer, Electrical, Automation, Control and Information Engineering*, *10*, 1780-1790.
[45] Bogdanovych, A., Rodríguez, J. A., Simoff, S., Cohen, A., & Sierra, C. (2009, May). Developing virtual heritage applications as normative multiagent systems. In *International Workshop on Agent-Oriented Software Engineering* (pp. 140-154). Springer, Berlin, Heidelberg.
[46] Vosinakis, S., & Avradinis, N. (2016). Virtual Agora: representation of an ancient Greek Agora in virtual worlds using biologically-inspired motivational agents. *Mediterranean Archaeology & Archaeometry*, *16*(5).
[47] De Paolis, L. T., Aloisio, G., Celentano, M. G., Oliva, L., & Vecchio, P. (2011, May). Experiencing a town of the middle ages: an application for the edutainment in cultural heritage. In *2011 IEEE 3rd International Conference on Communication Software and Networks* (pp. 169-174). IEEE.
[48] Shen, W., Hao, Q., Mak, H., Neelamkavil, J., Xie, H., Dickinson, J., ... & Xue, H. (2010). Systems integration and collaboration in architecture, engineering, construction, and facilities management: A review. *Advanced engineering informatics*, *24*(2), 196-207.
[49] Armas, D. (1509). Livro das Fortalezas Situadas no Extremo de Portugal e Castela por Duarte de Armas, Escudeiro da Casa do Rei D. Manuel I - Biblioteca Nacional Digital. Obtido de http://purl.pt/24908
[50] Lopes, V. (2018). O complexo religioso e os batistérios de Mértola na Antiguidade Tardia. *Medievalista. Online*, (23). Obtido de https://journals.openedition.org/medievalista/1570
[51] Macias, S., & Gonçalves, I. (1996). *Mértola islâmica: estudo histórico-arqueológico do Bairro da Alcáçova:(séculos XII-XIII)*.
[52] Yu, F., Lu, Z., Luo, H., & Wang, P. (2011). *Three-dimensional model analysis and processing*, (pp.20-21). Springer Science & Business Media.

[53] Saldana, M., & Johanson, C. (2013). Procedural modeling for rapid-prototyping of multiple building phases. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, *5*, W1.

[54] Cláudio, A. P., Carmo, M. B., de Carvalho, A. A., Xavier, W., & Antunes, R. F. (2017). Recreating a medieval urban scene with virtual intelligent characters: steps to create the complete scenario. *Virtual Archaeology Review*, 8(17), 31-41.

Anexo A: Abordagens da geração de terrenos

Este anexo complementa a secção 4.1, a construção do terreno real no Unity 3D e possui como objetivo informações e pormenores relativos as várias formas de gerar terreno assim como as principais ferramentas que são utilizadas.

A.1: Principais ferramentas pesquisadas responsáveis pela criação de terrenos.

Com o intuito de se determinar as principais ferramentas que existem para criação de terrenos, foi efetuada uma pesquisa, da qual resultou uma lista de várias ferramentas:

- *L3DT* (Bundysoft – Teste gratuito por 90 dias)[73];
- *Terragen 4* (Planetside – Versão gratuita / Creative / Professional)[74];
- *EarthSculptor* (Ernest Szoka)[75];
- *World Machine* (Stephen Schmitt – Gratuito / Standard / Professional)[76];
- *Fractscape* (StarsceneSoftware)[77];
- *Esri CityEngine* (Teste gratuito por 30 dias)[78];
- *Sketchup Pro* (Google Teste gratuito por 30 dias)[79];
- *Terrain.Party* (Web app que cria *heightmaps* com dados reais/ Gratuito)[80];

[73] L3DT, acedido em 8 de dezembro de 2018 em: http://www.bundysoft.com/docs/doku.php?id=l3dt:about

[74] Terragen 4, acedido em 8 de dezembro de 2018 em: http://planetside.co.uk/terragen-feature-tour/

[75] EarthSculptor, acedido em 8 de novembro de 2018 em: http://www.earthsculptor.com/

[76] World Machine, acedido em 18 de novembro de 2018 em: http://www.world-machine.com/

[77] Fractscape, acedido em 18 de novembro de 2018 em: https://starscenesoftware.com/fractscape.html

[78] Esri CityEngine, acedido em 18 de novembro de 2018 em: http://www.esri.com/software/cityengine

[79] Sketchup Pro, acedido em 18 de setembro de 2018 em: http://www.sketchup.com/products/sketchup-pro

[80] Terrain.Party , acedido em 18 de janeiro de 2019 em: http://terrain.party/

A.2: Manual da ferramenta terrain.party

A ferramenta *terrain.party* começa com o ecrã, centrado na Finlândia. É possível aumentar e diminuir o zoom utilizando os botões à esquerda, assim como alternar para o modo de ecrã inteiro com o botão à direita, de acordo com a Figura A.1.

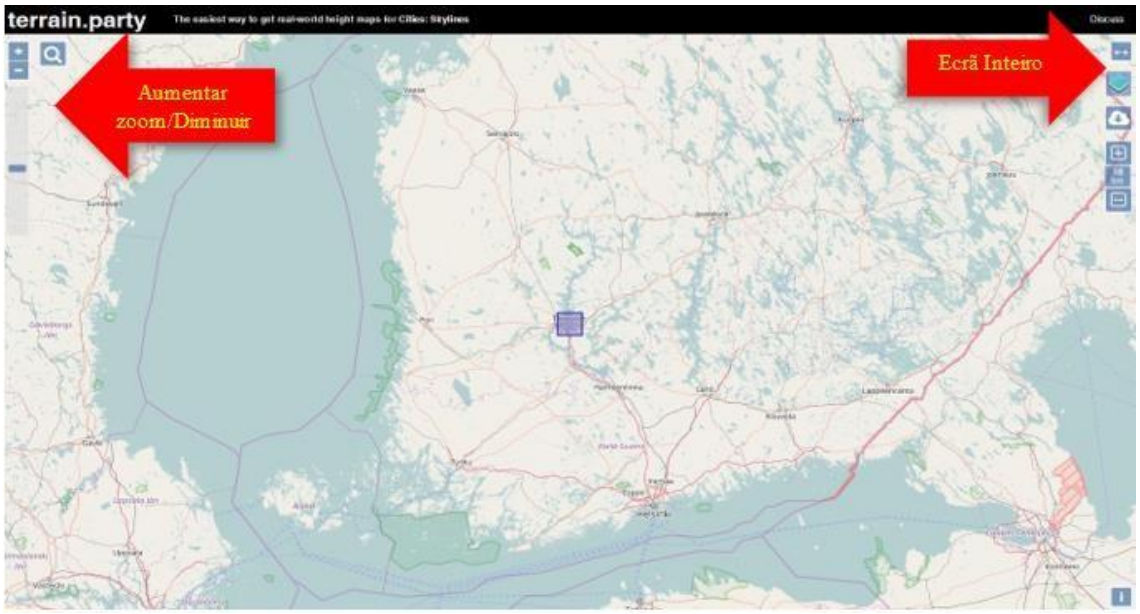


Figura A.1: Seleção de zoom e ecrã inteiro

Para selecionar o local pretendido, bastar clicar na lupa introduzindo o nome da cidade de acordo com a Figura A.2.

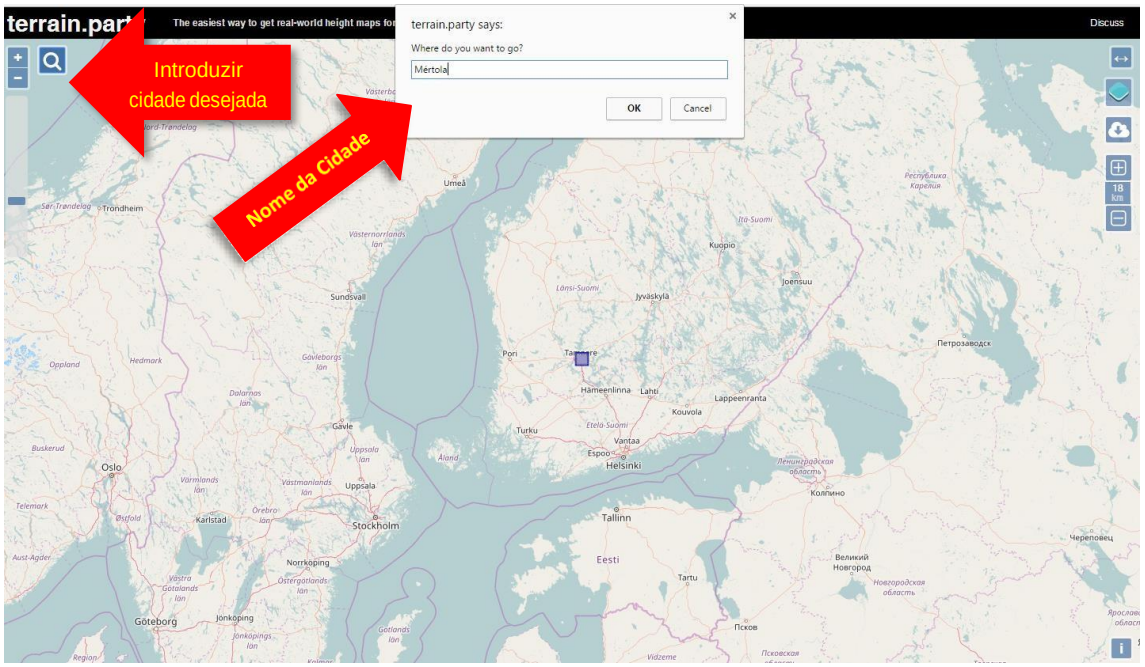


Figura A.2: Escolha da cidade desejada

Depois de selecionada a cidade é possível aumentar ou diminuir a área escolhida, para isso é necessário clicar nos botões + ou - de acordo com a Figura A.3, a seleção de cobertura da área escolhida varia entre 8 x 8 km até 60 x 60km.

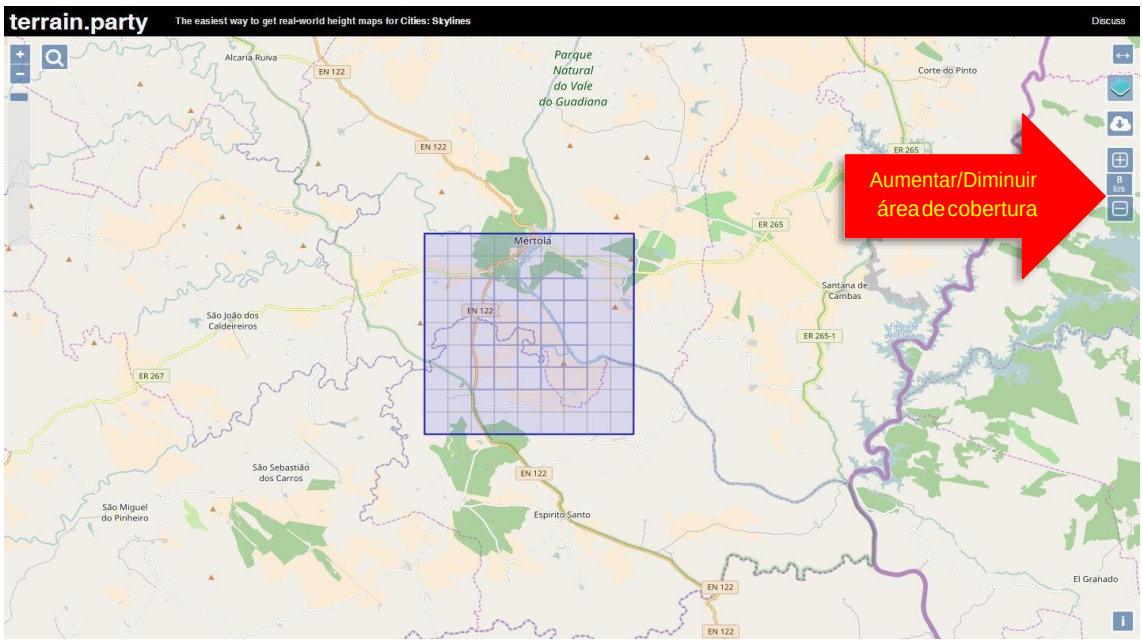


Figura A.3: Ajuste da área de cobertura.

Uma vez selecionada a cidade pretendida, assim como sua área de cobertura, o próximo passo é fazer o *download*, e o terrain.party solicitará um nome para o *heightmap* pretendido, de acordo com a Figura A.4.

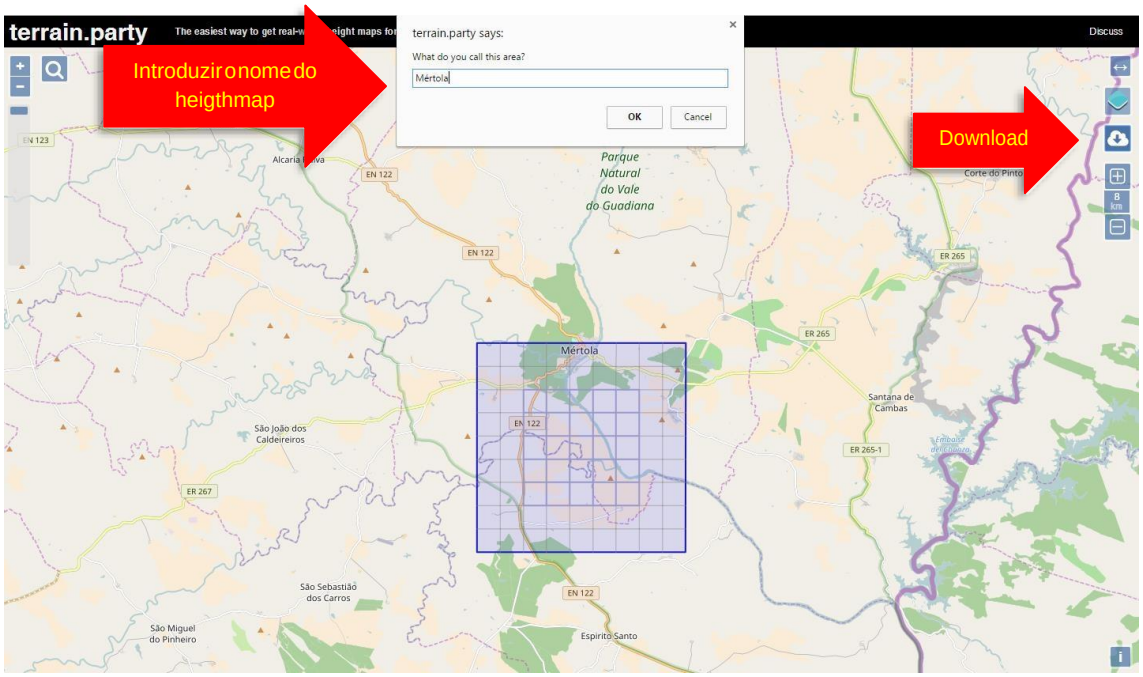


Figura A.4: Download do *heigthmap* desejado com seu respetivo nome.

A.3: Editor de terrenos do Unity 3D

Na escolha do pincel é possível escolher o seu tamanho, assim como a efetividade na deformação, a Figura A.5 mostra o editor de terrenos, e um exemplo de um terreno ajustado pelo editor de terrenos.

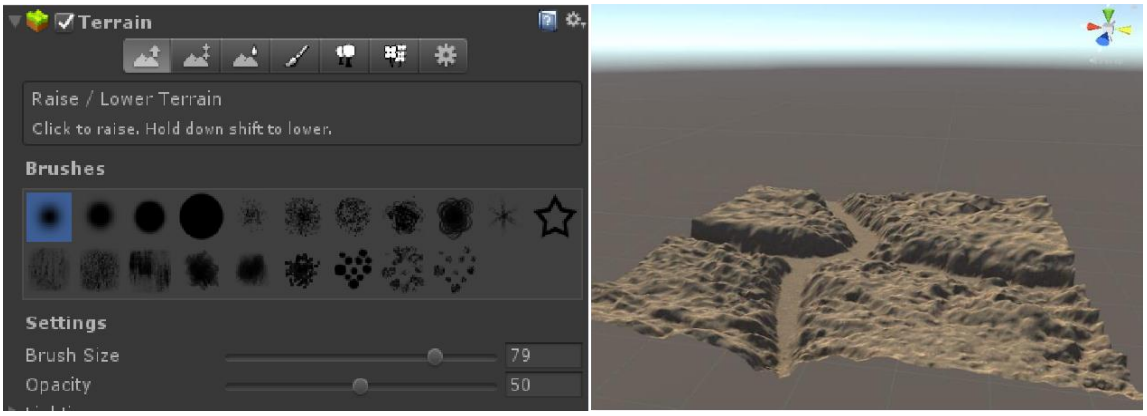


Figura A.5: À esquerda da figura temos as ferramentas do editor de terrenos e a direita um exemplo de um terreno ajustado pelo editor de terrenos.

- **Ferramentas de textura:**
  - **Textures**: Para utilizar uma textura é necessário atribuí-la ao terreno, para isso basta clicar no Edit textures>Add texture, sendo que a primeira textura é usada como base do terreno. Entretanto é possível acrescentar quantas texturas sejam necessárias, as subsequentes estarão disponíveis para pintar utilizando as ferramentas de pincel. Abaixo das texturas no *inspector* do terreno, é possível ver as opções habituais de tamanho e opacidade do pincel. Porém existe uma opção adicional chamada *Target Strength*, que configura o valor da opacidade máxima que o pincel irá aumentar, mesmo que ele passe pelo mesmo ponto repetidamente. Isto pode ser uma maneira útil de adicionar sutilmente pequenas manchas de cores variáveis dentro de um único tipo de terreno para quebrar a monotonia de uma grande área plana com a mesma textura que se repete inúmeras vezes. A Figura A.6 apresenta, à direita, a interface onde é possível selecionar, modificar ou apagar a textura desejada, a textura que será utilizada como base do terreno ficará na primeira posição do campo textures. Já a imagem à esquerda da Figura A.6 mostra a textura selecionada, assim como os parâmetros do tamanho.

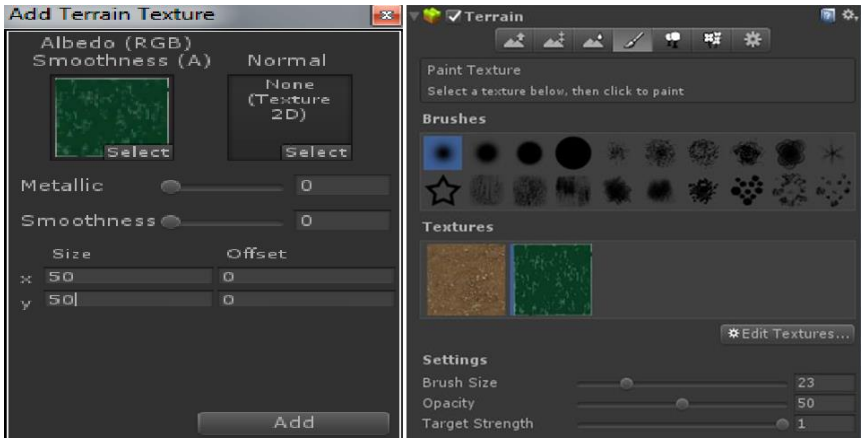


Figura A.6: Escolha da textura no editor de terrenos

• **Ferramentas de árvores:**

- **Trees**: Para adicionar as árvores basta clicar no *Edit Tree>Add Tree*, de acordo com a Figura A.8 onde posso selecionar o tipo de árvore desejada, sendo ele importada de outra ferramenta ou do próprio package padrão do Unity, é possível também a criação de milhares de árvores ao mesmo tempo através do *Mass Place Trees*, assim como ajustar de acordo com o necessário alguns parâmetros, a Figura A.7 apresenta as diferentes opções de ajustes das árvores no do editor de terrenos:
  - **Bend Factor**: fator de curvatura em relação ao vento;
  - **Brush Size**: número de árvores por clique;
  - **Tree Density**: proximidade das árvores;
  - **Tree Width/Height:** Tamanho das árvores;
  - **Mass Place Trees:** Criar automaticamente milhares de árvores de acordo com o tamanho do pincel.

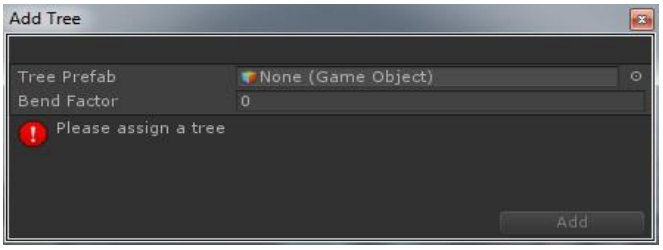


Figura A.8: Escolha do tipo de árvore no editor de terrenos

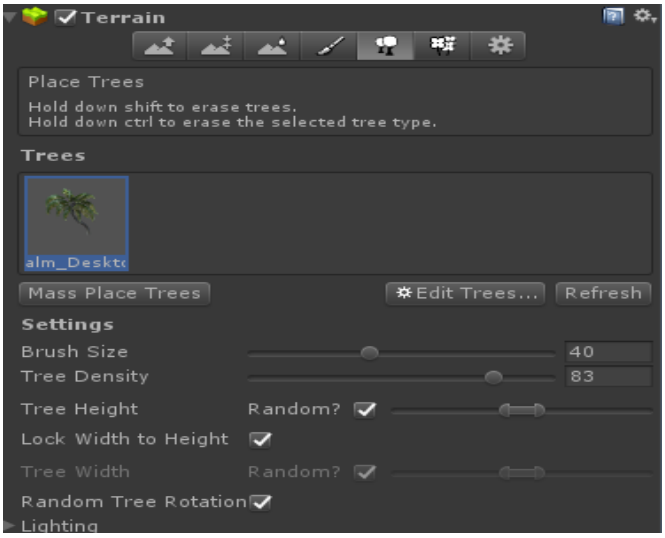


Figura A.7: Opções de ajustes das árvores no editor de terrenos.

• **Ferramentas de detalhes e relva:**

- Details: Aplica objetos de detalhe como relva, plantas, rochas. Na prática, é uma textura 2D com transparência ou 3D *low poly*. Para adicionar relva ao terreno é preciso clicar no *Edit Details >Add Grass*. Para adicionar objetos de detalhe como rochas é preciso clicar no *Edit Details >Add Detail Mesh*.
  - **Detail Texture**: textura utilizada para a relva;

- **Min**/ **Max Width** / **Height**: largura/ altura mínima/ máxima em metros,
- **Noise Spread**: fator de agrupamento;
- **Healty Color**: cor predominante do centro da relva;
- **Dry Color**: cor predominante nos cantos;
- **Grayscale Lightning:** desabilita a iluminação da relva por luz colorida;
- **Bilboard:** A relva vira em direção à câmara.

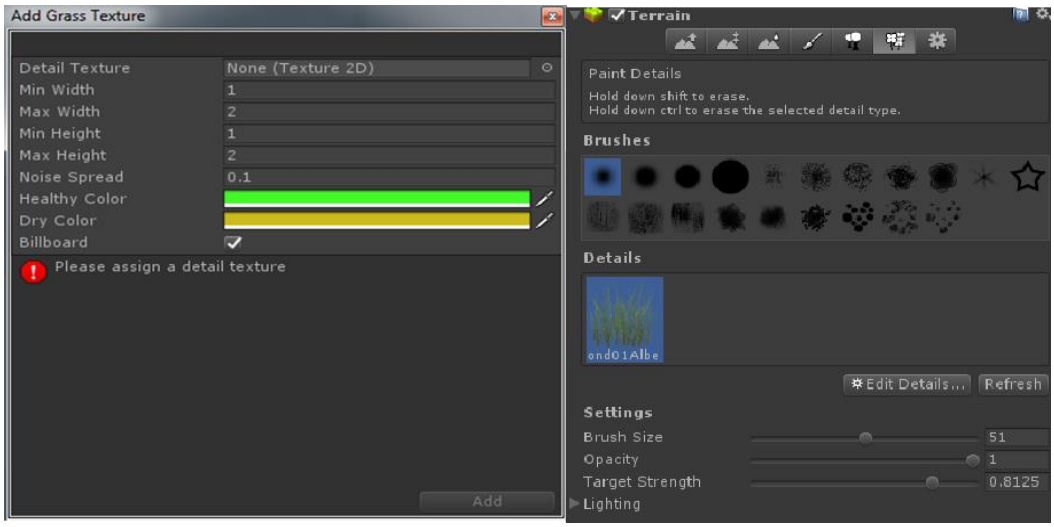


Figura A.9: Opões de adição e ajustes da relva no editor de terrenos.

Zonas de Vento: Cria efeitos de ventos reais, nas árvores colocadas no terreno.

- **Adicionar zona de vento**: *GameObject > 3D Object >Wind Zone.*
- **Propriedades:**
  - **Mode**: (Spherical ou Directional);
  - **Radius**: raio;
  - **Main**: força do vento;
  - **Turbulence**: variação na força do vento;
  - **Pulse Magnitude**: variação do vento ao longo do tempo;
  - **Pulse Frequency**: frequência das mudanças do vento;

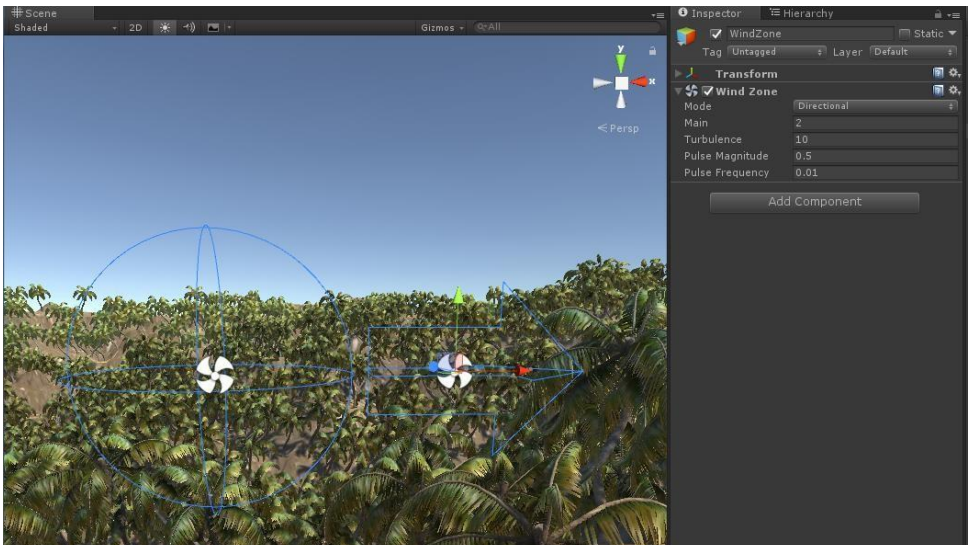


Figura A.10: Propriedades do vento.

- **Configurações do Terreno:**
  - **Base Terrain:**
    - **Draw:** habilita/desabilita a renderização do terreno;
    - **Pixel Error**: precisão do mapeamento da estrutura do terreno (mapa de altura, texturas, etc.) e o terreno gerado;
    - **Base Map Distance:** distancia máxima em que as texturas do terreno são renderizadas em resolução máxima;
    - **Cast Shadows**: habilita/desabilita sombras no terreno;
    - **Material:** material utilizado na renderização do terreno;
    - **Thickness:** extensão do volume de colisão;
  - **Tree & Detail Objects:**
    - **Draw**: habilita/desabilita a renderização de árvores e detalhes do terreno;
    - **Detail Distance**: distancia máxima em que os detalhes são renderizados;
    - **Detail Density**: número máximo de detalhes em uma única unidade de área;
    - **Tree Distance**: distancia máxima em que as árvores são renderizadas;
    - **Billboard Start**: distancia em que as árvores são substituídas por imagens billboards;
    - **Fade length**: distancia na qual as árvores vão transitar entre modelos 3D e billboards;
    - **Max Mesh Trees**: número máximo de árvores representadas por objetos 3D;
  - **Resolution:**
    - **Terrain Width**: tamanho do terreno no eixo x;
    - **Terrain Length**: tamanho do terreno no eixo z;
    - **Terrain Height**: altura do terreno;
    - **Heightmap Resolution**: resolução do mapa de alturas do terreno usado para armazenar a topologia. Valores de $(2^n + 1)$;
    - **Detail Resolution**: Resolução do mapa para armazenar árvores, rochas, etc. Quanto maior, mais preciso o posicionamento;
    - **Control Texture Resolution:** Resolução do mapa utilizado para posicionar as texturas pintadas no terreno (*Splatmap*), controla o detalhamento das texturas;
    - **Base Texture Resolution**: Resolução da textura usada em lugar do *splat map* depois de uma certa distância**.**

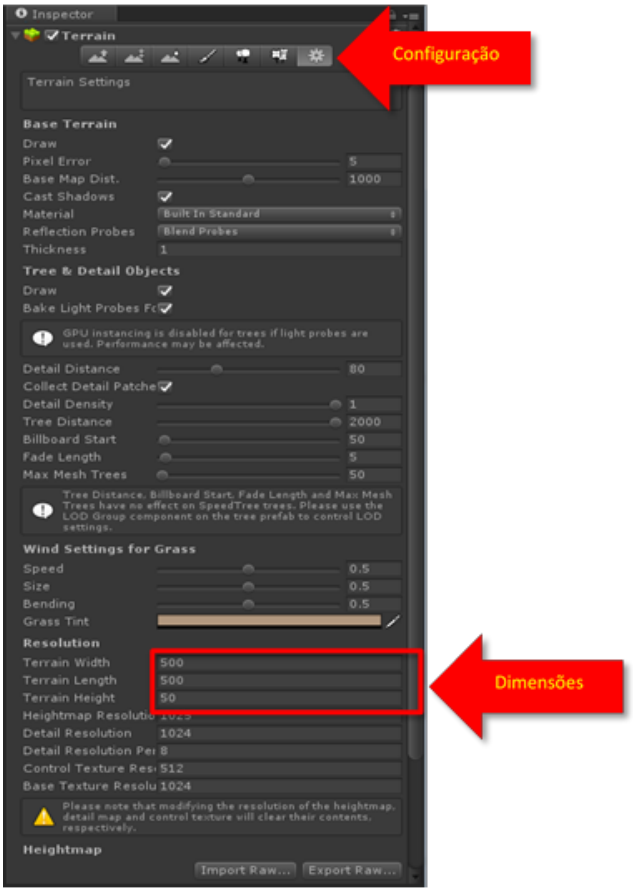


Figura A.11: *Inspector* de um objeto *Terrain.*

De acordo com a Figura A.11, onde podemos configurar o terreno mediante os parâmetros mencionados, cabe destacar a importância do parâmetro, *Resolution***,** na qual devemos configurar as dimensões do terreno, antes do *heightmap* ser importado através do *Import Raw* com isso podemos determinar o tamanho do terreno no eixo x e z, assim como sua altura.

A.4 Passos para importar um heightmap para Unity 3D através de script

O primeiro passo que se deve fazer para utilizar o *script* é realizar o *download* na página *Wiki* do Unity 3D mencionada anteriormente. Uma vez realizado o *download* é preciso abrir o Unity e importar o *script*, para fazê-lo, basta arrastar o *script* para os assets do projeto. Com isso o *script* será importado como uma extensão da interface do editor do Unity, depois disso, é preciso criar um objeto do tipo terreno, clicando com o botão direito no *hierarchy* e selecionar *3D Object >Terrain*, isso fará com que o terreno seja criado.

Por padrão, o terreno será 500x500 é necessário alterar o tamanho para o valor desejado, sendo atribuído as dimensões dadas na criação do *heightmap* que será importado para que o terreno fique harmonioso e proporcional às vezes devido a conversão dos valores de altura e largura poderá ser necessário realizar pequenos ajustes.

Uma vez criado o terreno com o *script* já importado, o segundo passo será importar o respetivo heightmap em formato .png para o Unity 3D. A maneira de fazê-lo será análoga ao primeiro passo mencionado anteriormente, basta arrastar o heightmap para os assets do projeto, como o heightmap é uma imagem formato .png é preciso torná-la legível para o *script*, para isso é preciso ir aos assets do projeto e buscar a imagem importada selecionando-a, em seguida , no *inspector* onde diz *Texture Type*, selecione *Advance > Read/Write Enabled >Apply*.

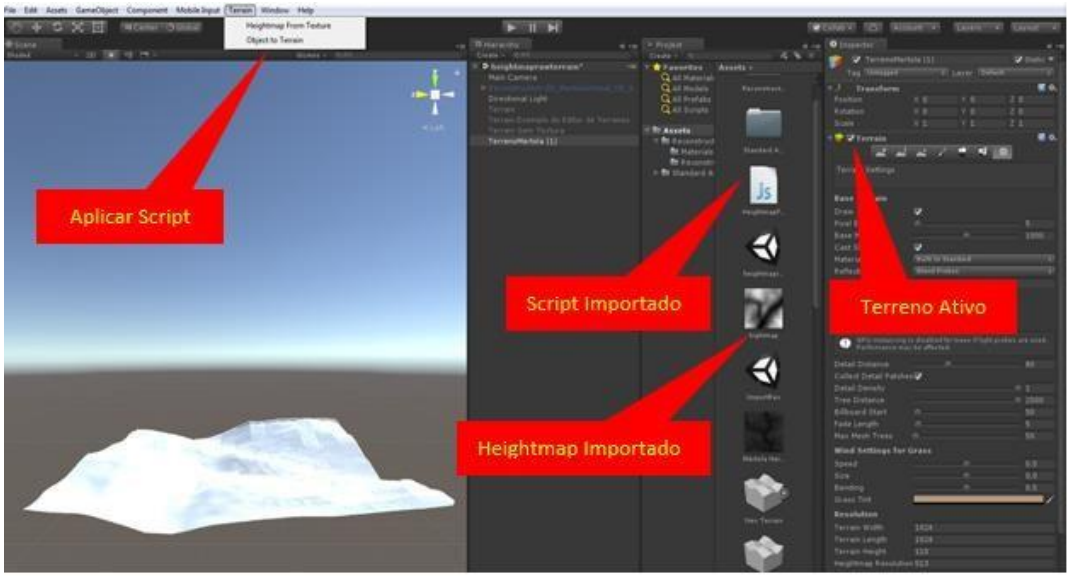


Figura A. 12: Importação de um heightmap através de um *script*.

Depois de ativar o Read/Write do *heightmap*, é preciso selecioná-lo nos assets do projeto, uma vez selecionado, o próximo passo será aplicar o *script*, para isso, é imprescindível ir na barra de ferramentas localizado na parte superior e clicar em *Terrain >Heightmap From Texture*. Com isso teremos criado um terreno real a partir de um *heightmap*, o *script* em si é bastante simples e irá editar todo o terreno em uma cena, portanto, ao aplicar o *heightmap*, é fundamental certificar- se de que exista apenas um terreno definido como ativo.

A.5 Passos para geração de terreno através da ferramenta Sketchup Pro

A Figura A.13 mostra o passo onde é introduzido a coordenada (37.633, -7.642) no Google SketchUp sendo necessário selecionar o terreno desejado assim como seu tamanho.

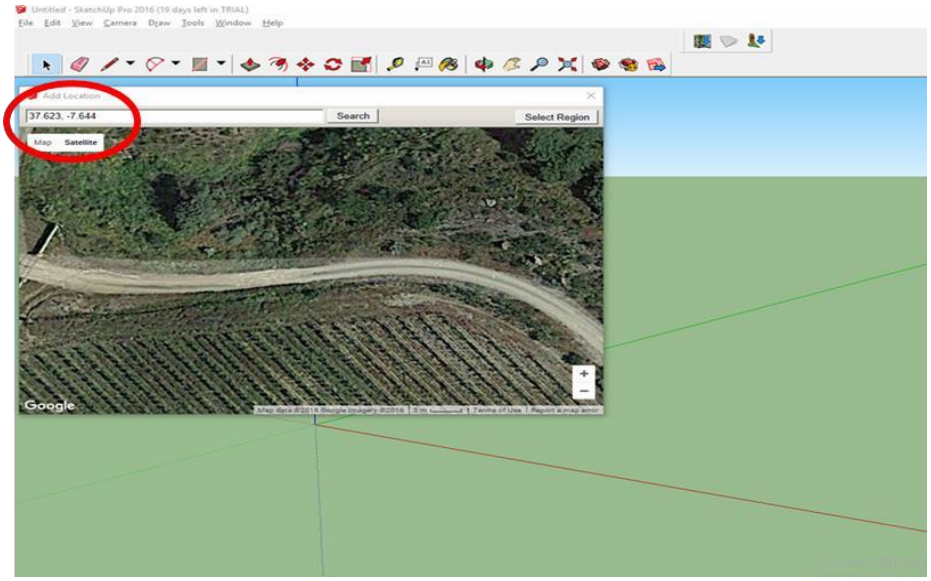


Figura A.13: Coordenada introduzida para seleção do terreno.

Ao contrário do Google Earth que nos dá exatamente o ponto certo com a coordenada específica, uma particularidade no Google SketchUp é que ao introduzir a coordenada, nos dá

uma região inteira, sendo necessário uma confirmação visual com Google Earth, no intuito de confirmar com precisão aonde fica a região escolhida. Depois de selecionada a região que pode ser aumenta num raio de até 2 km, é necessário fazer o *Grab*. A região escolhida é apresentada de forma *flat*, sendo necessário executar a ação *toggle terrain* da opção no menu de botões do Google SketchUp.

No segundo passo é preciso selecionar a área do terreno desejado e fazer o *Grab*. A Figura A.14 ilustra esse passo.

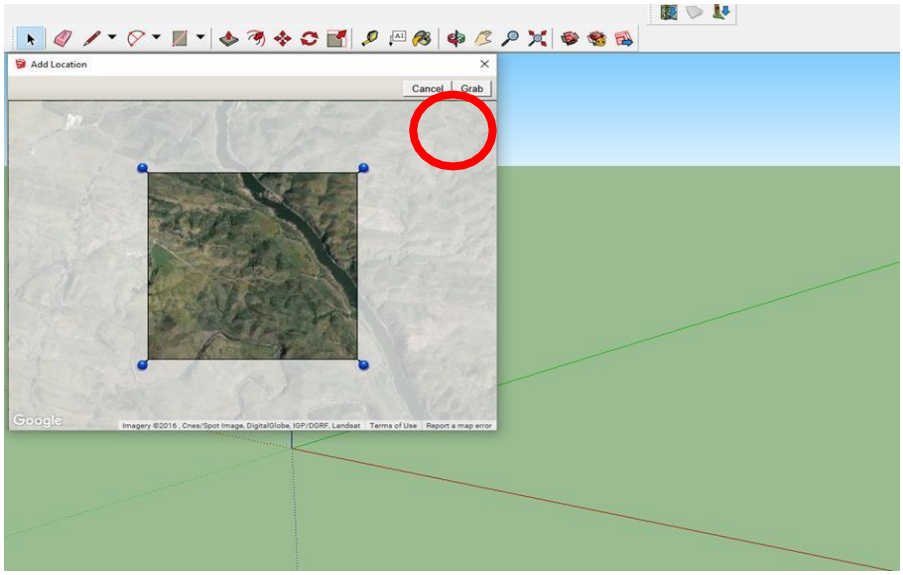


Figura A.14: Seleção do tamanho do terreno para o corte.

Uma vez feito o *Grab* da região escolhida, ela é mostrada no ecrã de acordo com a Figura A.15, nota-se que o terreno mostrado é plano.

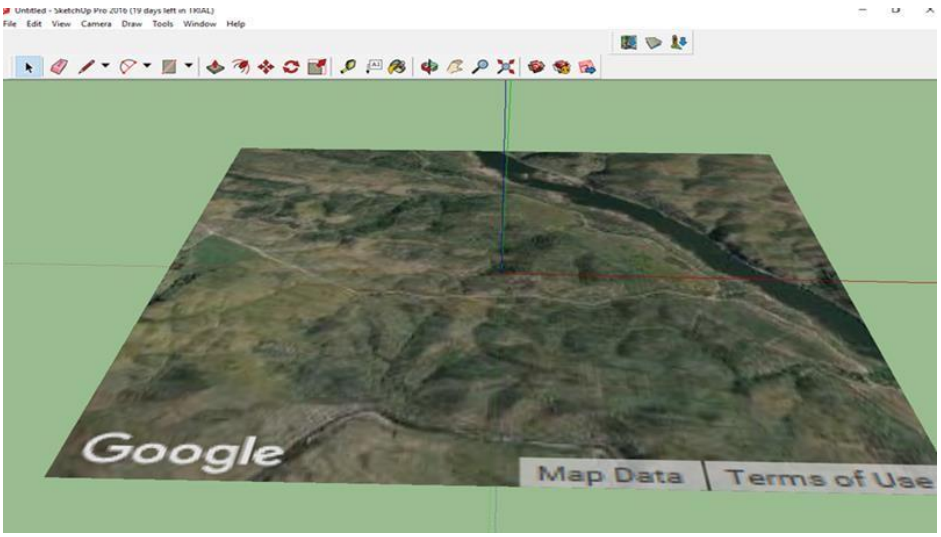


Figura A.15: Terreno escolhido.

Depois de escolhida a região é necessário fazer o *toggle terrain* para que o terreno ganhe forma "real" com suas elevações, a Figura A.16 mostra o procedimento mencionado.

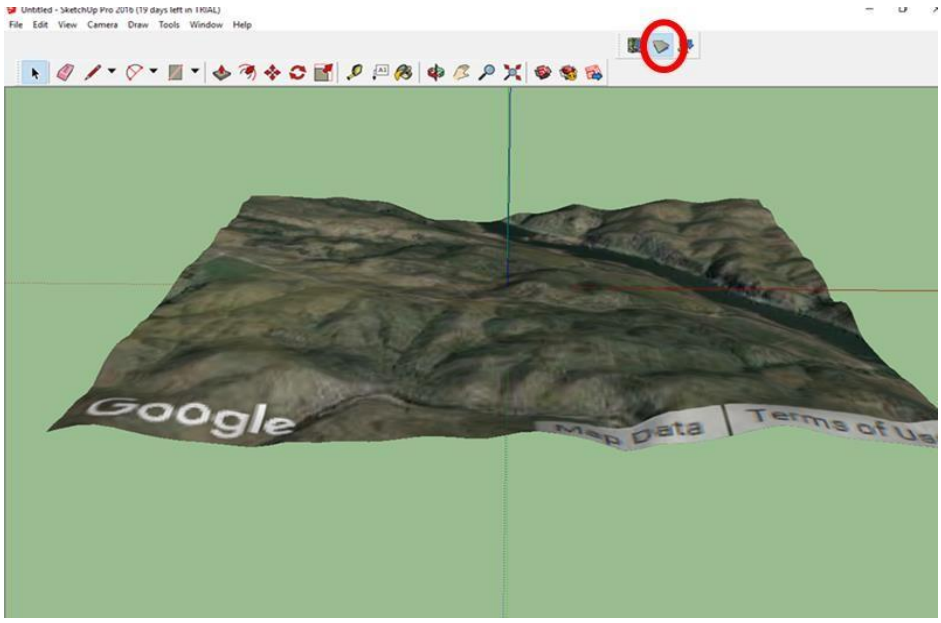


Figura A.16: Terreno na forma real com suas elevações.

Como é possível observar, na Figura A.15 temos o região selecionada de acordo com as coordenadas desejadas (37.633, -7.642), Por omissão todos os terrenos no Google SketchUp, depois de realizado o "Grab" são apresentados como uma *mesh* plana, sendo necessário aplicar a funcionalidade *toggle terrain* para que apresente as elevações reais do terreno inicialmente escolhido, às figuras A.15 e A.16 representam as respetivas diferenças.

Uma vez feito o procedimento para a coordenada (37.633, -7.642), é necessário repetir os mesmos passos para todas a coordenadas desejadas, sendo uma de cada vez e selecionando a região até obter o terreno desejado.

A.6: Passos para exportar um ficheiro da ferramenta SketchUp para o Unity 3D.

1. Salvar o ficheiro do SketchUp em formato .FBX. *File* > *Export* > *3D Model* > *Export Type*: .FBX (Provavelmente é possível importar do SketchUp para o Unity salvando o ficheiro do SketchUp em outro formato, porém fizemos os testes utilizando o formato .DAE, ficando muito pesado no Unity).
2. Importação do SketchUp para o Unity 3D: Abra o Unity e importe o ficheiro do SketchUp na pasta *Assets*. Pode-se também arrastar o ficheiro do SketchUp diretamente para a pasta *Assets*, no Unity. Ao salvar o ficheiro do SketchUp no formato .FBX, uma pasta é criada com os materiais usados no modelo. Para que esses materiais sejam aplicados no Unity, deve-se copiar essa pasta e colá-la na pasta *Assets* do Unity (ou arrastá-la diretamente para a pasta).

Uma vez importado para o Unity 3D, surgiram algumas questões a serem levadas em consideração.

a. O encaixe será perfeito?
b. Será necessário converter a nova *mesh* importada em *terrain* tal como a cidade?
c. Qual seriam as vantagens de termos 2 *terrains*, em termos de realismo e qualidade da imagem no entorno da cidade?
d. O que seria melhor em termos de desempenho, o entorno convertido a *terrain* ou *mesh*?

Para isso foram realizados os seguintes testes:

A *mesh* gerada pelo Google SketchUp teria que encaixar perfeitamente, já que as coordenadas são reais, porém para isso teríamos que verificar tanto no projeto antes de ser convertido para terrain como no projeto convertido. Portanto primeiro foi testado no ambiente do Unity 3D o projeto final realizado pelo Alexandre Carvalho [3], já que nele o terreno da cidade de Mértola era uma mesh e o entorno gerado e importado do Google sketchUp também era uma *mesh*.

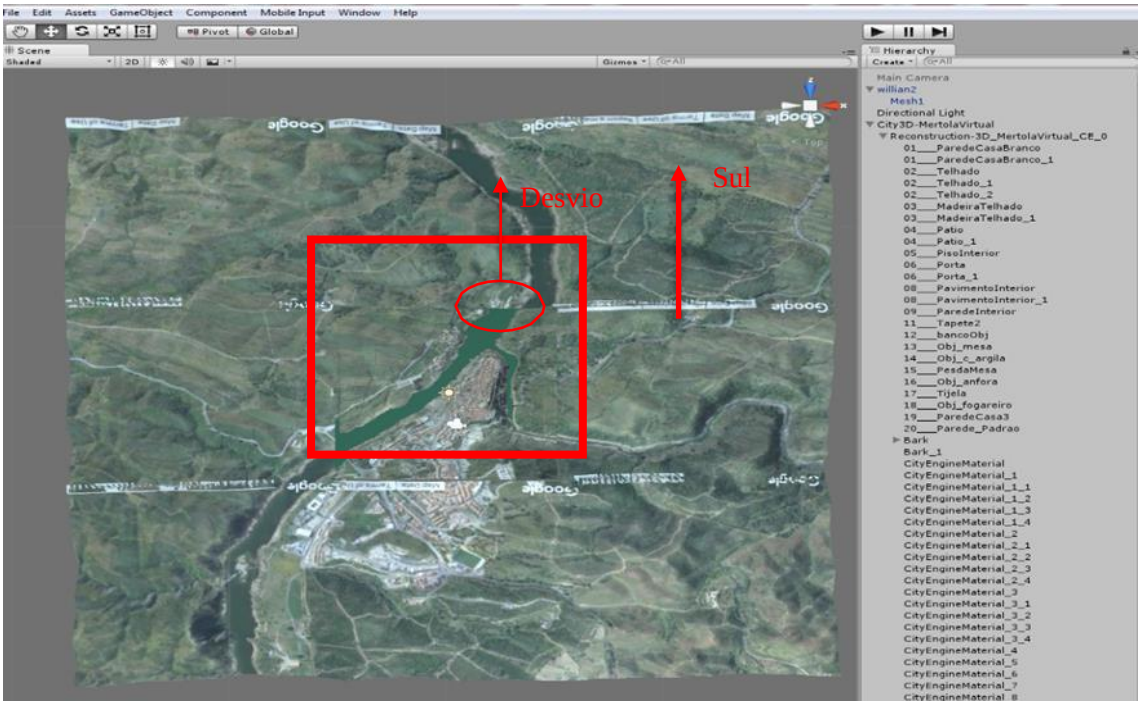


Figura A.17: Teste realizado com a mesh do Google Sketchup (região externa) com a mesh (região interna) da cidade de Mértola.

A Figura A.17 mostra o teste realizado com a *mesh* gerada pelo Google SketchUp e importada para o Unity 3D (terreno "extra") com a *mesh* da cidade de Mértola inicialmente importada também para o Unity.

Como era esperado o encaixe foi quase perfeito, nota-se um pequeno desvio no curso do rio ao sul na parte de cima da Figura A.17 apesar de termos introduzido as mesmas coordenadas, ao serem introduzidas no Google SketchUp, foi necessário realizar um pequeno ajuste manual já que ele não mostra especificamente a região exata, sendo assim necessário fazer uma aproximação de acordo com Google earth, foram tomadas referências das coordenadas de aproximadamente 1 km, das coordenadas utilizadas no projeto do Alexandre Carvalho[3].

Como referido na secção 4.2.1 a cidade de Mértola foi convertida de *mesh* para *terrain*, diante dessa premissa o próximo passo então, seria saber se a *mesh* gerada Google SketchUp e importada para o ambiente Unity 3D, se encaixaria perfeitamente no *terrain*. Para isso foi feito o teste de acordo com a Figura A.18, onde é ilustrado a *mesh* parte externa e o *terrain* a parte interna.

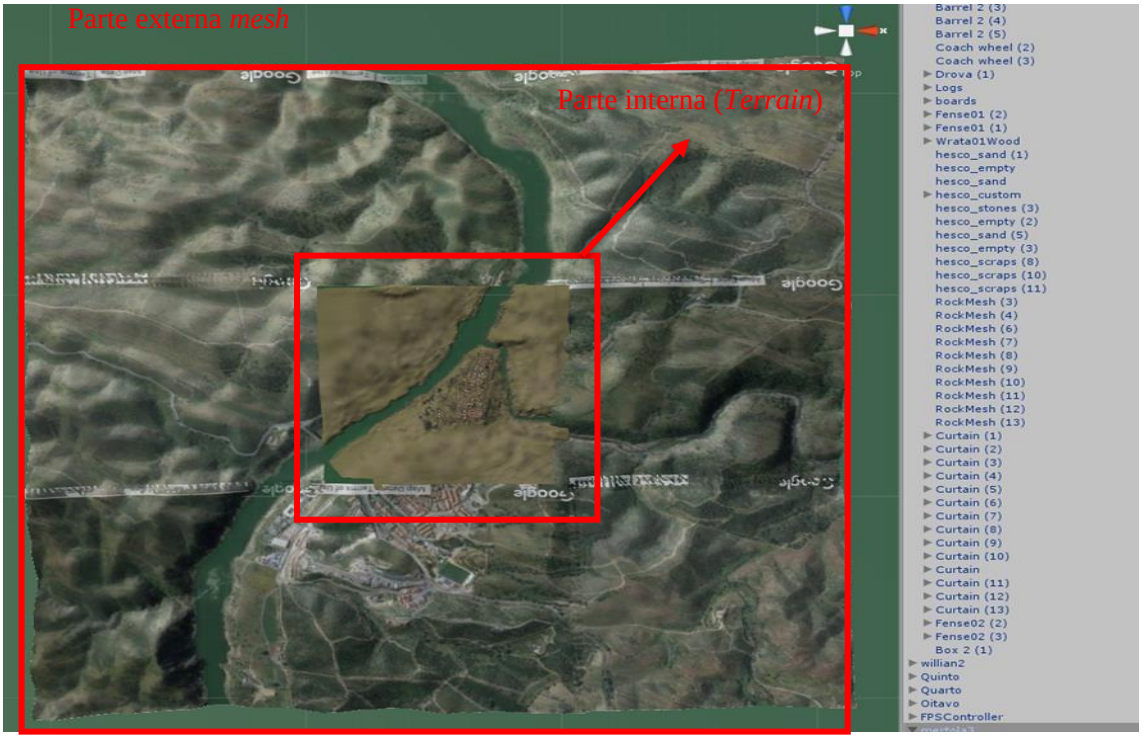


Figura A.18: Teste realizado com a *Mesh* do Google SketchUp (região externa) com a *mesh* já convertida a *terrain* (região interna) da cidade de Mértola.

Como era esperado a *mesh* não encaixou perfeitamente, devido aos ajustes feitos depois da conversão da *mesh* original para *terrain* da cidade de Mértola, além disso como no primeiro teste, mostrado na Figura A.17 (*mesh* parte externa + *mesh* parte interna) também não havia sido encaixado na sua perfeição, esperava-se um comportamento análogo neste teste, apesar do pequeno ajuste realizado na junção da parte externa com a interna, já que a parte interna (*terrain*) foi possível utilizar o editor de terrenos, onde foram feitos alguns ajustes pontuais no intuito de suavizar as diferenças entre as duas partes, porém como a parte externa era uma *mesh*, onde não é possível tirar proveito do editor de terrenos, com isso a junção final acabou por ter muitos *gaps* ou seja buracos, que eram visíveis. Como a parte interna já tinha sido convertida para *terrain* e pensando em termos de facilitar a integração das duas regiões, foi ponderado a conversão da região externa para *terrain.*

Se compararmos em termos de qualidade e eficiência em relação a conversão ou não de um terreno que inicialmente é importado como *mesh* para *terrain*, tudo irá depender de que forma o terreno irá ser utilizado e em que condições, de acordo com o objetivo proposto, seria necessário ter um terreno mais simples possível devido sua finalidade, no que seria utilizado para "compor" a parte que faltava e considerando também que a sua superfície não seria navegável, ou seja, não teríamos personagens virtuais a andar nesta parte. Neste caso teríamos premissas suficientes para considerar a não conversão do terreno importado para *terrain*, porém como foi demonstrado na Figura A.18 a parte interna e a externa do terreno não encaixaram bem, essa possibilidade foi descartada.

Pensando em tirar proveito das ferramentas do editor de terrenos onde é possível ajustar o tamanho das montanhas desde quando seja necessário e com isso, "suavizar" o encaixe dos terrenos, neste contexto o entorno foi convertido para *terrain* no intuito de termos 2 *terrains* (parte interna e parte externa) e assim, proceder para o encaixe não sendo necessário realizar grandes ajustes.

A seguir, é demonstrado os passos que foram feitos para a conversão da *mesh* (parte externa) para *terrain*. O passo a passo é análogo ao explicado anteriormente na secção 4.2.1 onde a parte interna foi convertida, porém neste passo a passo existem algumas diferenças.

Para executar a conversão é necessário utilizar o *script object2terrain*. Porém na versão do *script* utlizada, foi necessário criar um *terrain* na plataforma Unity 3D *GameObject > 3D Object > Terrain.*A Figura A.19 mostra os passos iniciais para a conversão da *mesh*.

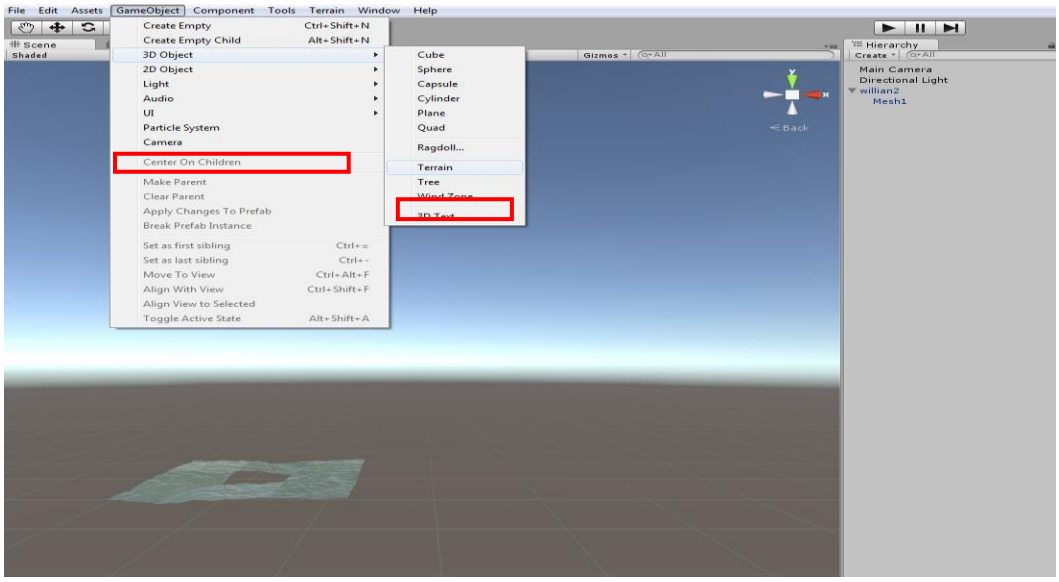


Figura A.19: Conversão da *mesh* importada do Google SketchUp para *terrain*.

Como podemos observar temos a *mesh* importada do Google à esquerda para que na direita seja colocado o novo *terrain*, como é ilustrado na figura abaixo:

Uma das premissas na criação de um terreno e ter atenção aos parâmetros de largura altura

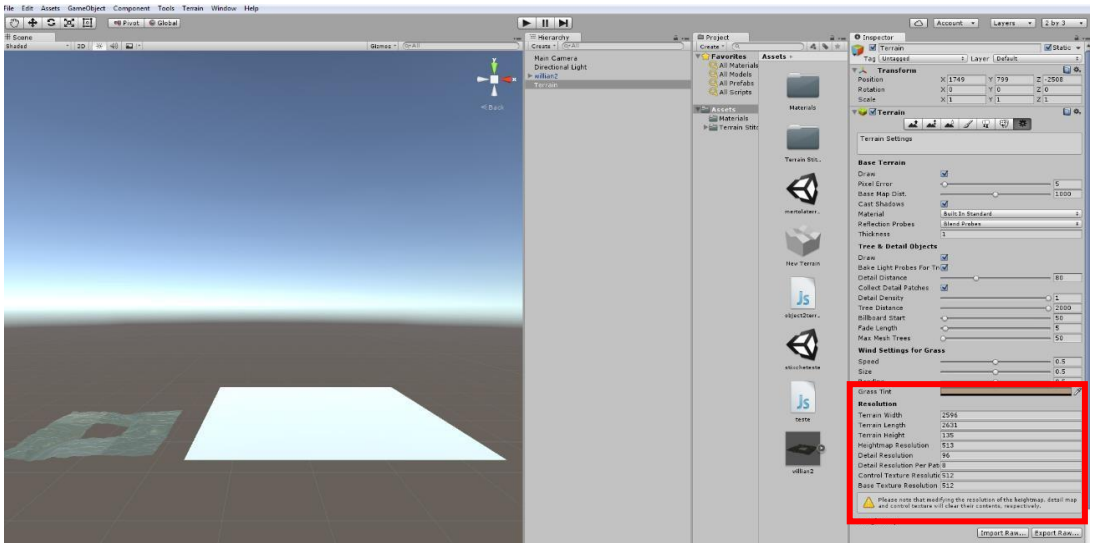


Figura A.20: *Mesh* importada do Google SketchUp e *terrain* pronto para ser convertido.

e comprimento, para o nosso caso os parâmetros foram: 2596, 135, 2631. Estes valores foram estabelecidos a partir de uma mudança de escala baseada no *terrain* convertido que originou na cidade de Mértola, que foram 1081.572,101.485,1096.417, primeiramente multiplicou-se e por 3 obtendo um valor aproximado de 3245,101.5, 3289, porém com estes valores o terreno exterior sobrava bastante. Mediante esse problema optou-se por tentar reduzir a escala, e os valores mais próximos do ótimo foram resultante da mudança de escala em 0.8.

Uma vez aplicado os valores, de acordo com a Figura A.20, é possível identificar na parte destacada em vermelho, o passo seguinte é aplicar o *script* de conversão para isso é necessário pressionar a opção *Object to Terrain*, *Terrain* > *Object to Terrain*. Uma vez executado os passos anteriores o terreno é criado de acordo com a Figura A.21.

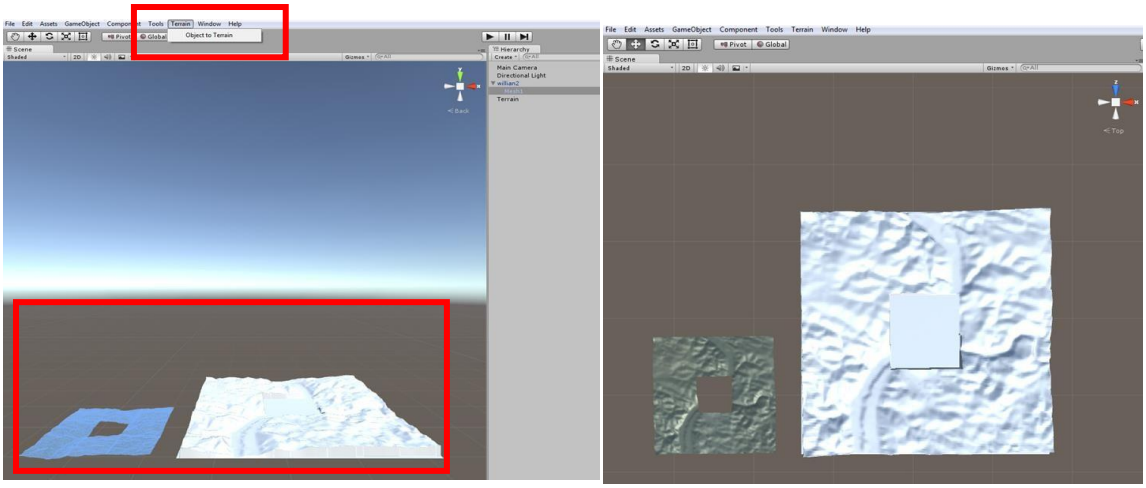


Figura A.21: Mesh do Google skecthUp a esquerda e o terrain convertido a direita no Unity depois de aplicado o *script* de conversão. Na parte da direita temos a vista top view.

Depois de convertido, o próximo passo foi proceder ao encaixe da cidade de Mértola com o entorno convertido a *terrain*, como mostrado na Figura A.22.

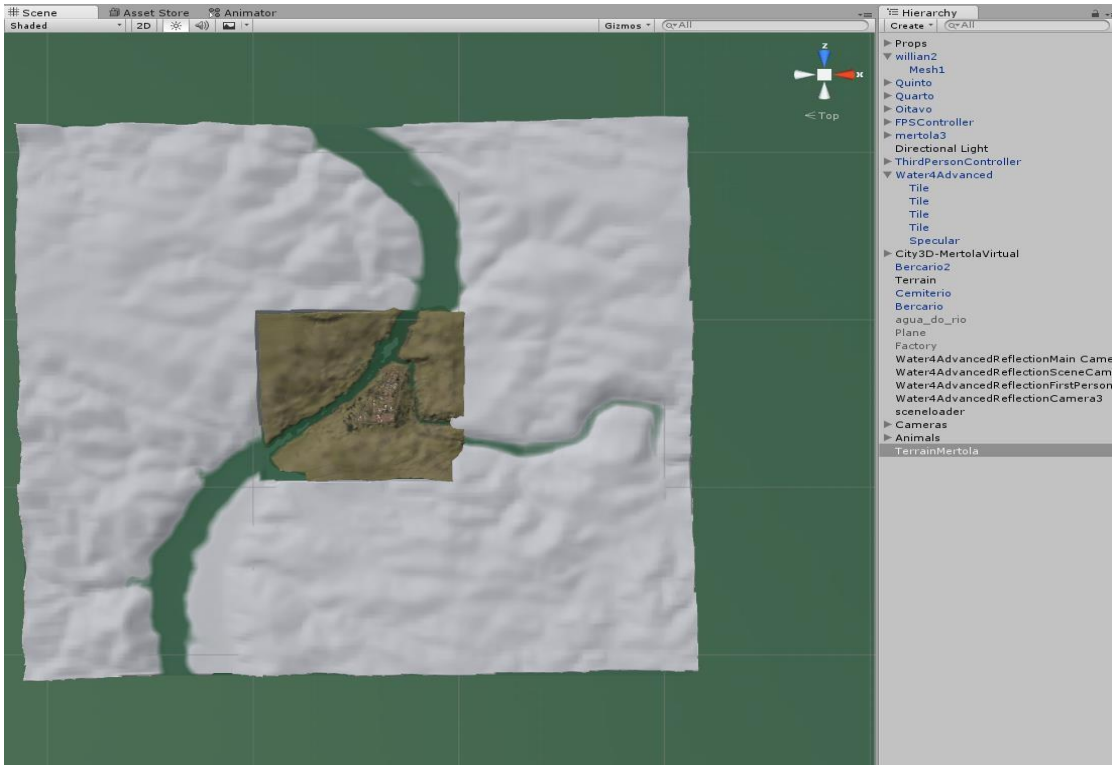


Figura A.22: *Terrain* recentemente convertido sem textura (região exterior) encaixada no *terrain*.

De acordo com a Figura A.22, e como já era esperado o encaixe não foi perfeito, sendo necessário realizar alguns ajustes no *terrain*, mediante isso foi realizado um pequeno corte para que o terreno se ajustasse de forma harmónica, alinhando ambas partes do rio Guadiana, tanto no sul como norte, foi diminuída também a largura do rio através das ferramentas do editor

de terrenos, além disso foi acrescentada uma textura para uma melhor visualização. Na Figura A.23 é possível identificar a parte do “encaixe” com sua respetiva textura, assim como o ajuste que foi feito para que o terreno ficasse mais harmónico possível.

Figura A.23: *Terrain* (parte interna) encaixada no *Terrain* ajustado com sua respetiva textura (parte externa).

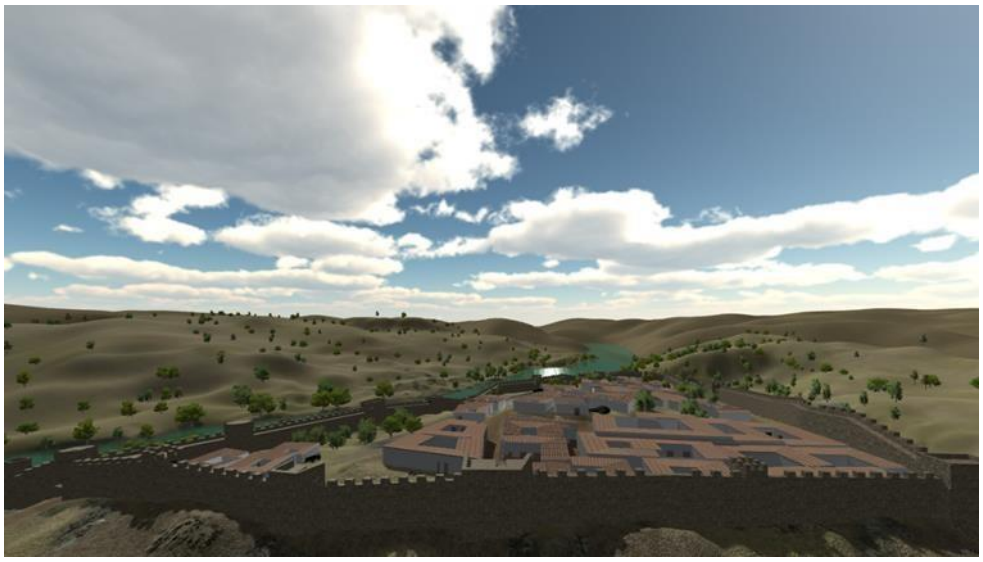


Figura A.24: Cidade de Mértola integrada ao entorno

A Figura A.24 mostra a cidade de Mértola integrada de forma harmoniosa com seu entorno.

A.7: Terrain SetNeighbors

Quando temos 2 ou mais terrenos e queremos juntá-los num intuito de se criar um terreno grande no Unity 3D existe uma função chamada *terrain.setneighbors*, que é utilizada para configurar a conexão dos terrenos adjacentes estabelecendo uma ligação entre eles, mas para isso é extremamente importante que as alturas dos terrenos (*heightmaps*) correspondam exatamente ao longo das bordas, já que essa função não resolve as diferenças de alturas entre terrenos.

Assumindo que as alturas sejam as mesmas em ambos lados que irão fazer a ligação é necessário especificar em cada terreno qual irá juntar-se a direita, esquerda, *top* e *bottom, isto é,* referindo-se as indicações vistas no na *scene* do Unity, como por exemplo: –X (esquerda) , +X(direita), Z+(topo), -Z (parte inferior).

Em termos genéricos se tivermos 9 terrenos onde cada número de 0 a 8 corresponde a um terreno, com isso teríamos uma matriz de 3x3 cujo seu centro corresponderia ao terreno número 4 tendo então 4 vizinhos: 7 (top), 3 (left), 5 (right), 1 (bottom).

6 **-7-** 8
3 -4- **5**
0 **-1-** 2

Para especificar os vizinhos para o motor de terreno, é necessário que criar um *script* em *java script* ou c#.

```
1  #pragma strict
2
3  var terrainLeft : Terrain;
4  var terrainTop : Terrain;
5  var terrainRight : Terrain;
6  var terrainBottom : Terrain;
7
8
9
10 function Start () {
11
12  var thisTerrain : Terrain = GetComponent (Terrain);
13  thisTerrain.SetNeighbors (terrainLeft,  terrainTop, terrainRight, terrainBottom);
14
15 }
```

Figura A.25: *Script* em javascript para utilizar a função terrain.setneighbors.

Para utilização este *script* é necessário colocá-lo no terreno que corresponde à posição 4, passando as referências das posições dos seus vizinhos 7,3,5,1 de acordo com a matriz explicada anteriormente. Isso é feito no *inspector* dentro do Unity na referência do terreno.

A Figura A.26 mostra a aplicação do *script* onde os terrenos são colocados de acordo com suas respetivas posições.

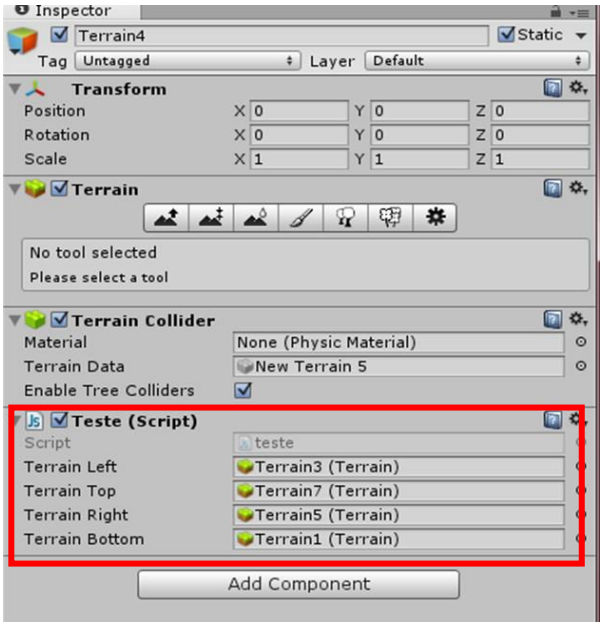


Figura A.26: Adição dos terrenos correspondentes aplicando o *script*.

De acordo com mostrado na figura acima, uma vez referenciado os terrenos nas suas posições, o *script* criar as referências dos vizinhos para o terreno no centro da matriz, contudo a fim de funcionar corretamente, os seus vizinhos devem ter referências recíprocas apontando para esse terreno, neste caso as conexões vizinhas são bilaterais (*2way*). Com este *script* podemos colocar em cada variável (objeto) um terreno, unicamente é necessário atribuir os links manualmente. No caso em que terrenos não tenham um vizinho em um determinado sentido, é só deixar a variável vazia na aba do *inspector*.

Na Figura A.27 temos um exemplo onde temos 5 terrenos em que suas Lods *(Level of Details*) foram interligadas através do *script*.

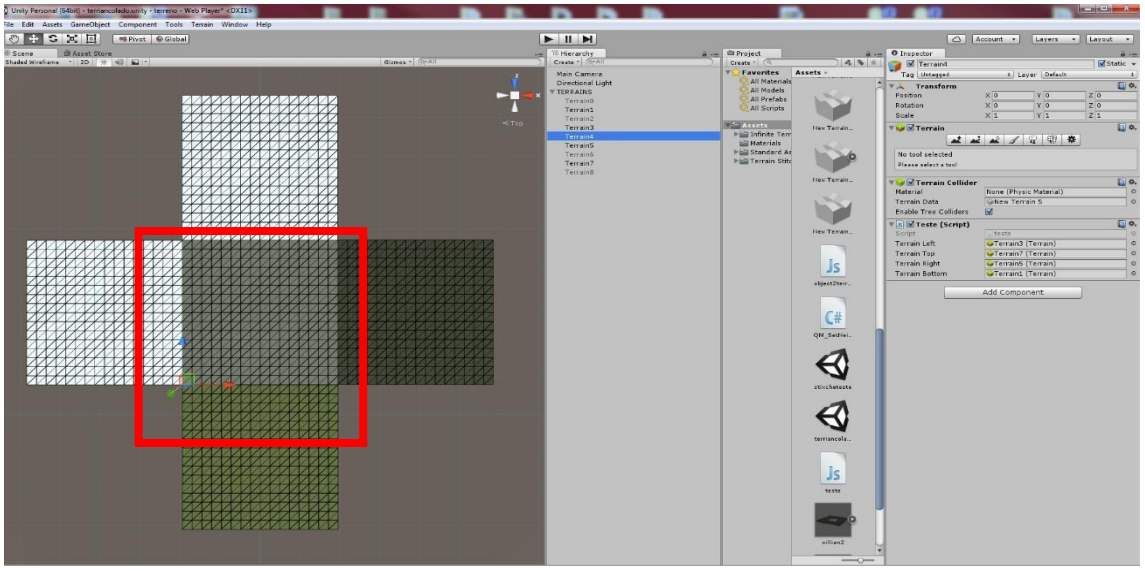


Figura A.27: Terreno interligado.

Existe uma variante do *script* anteriormente citado, mas agora na linguagem c#.

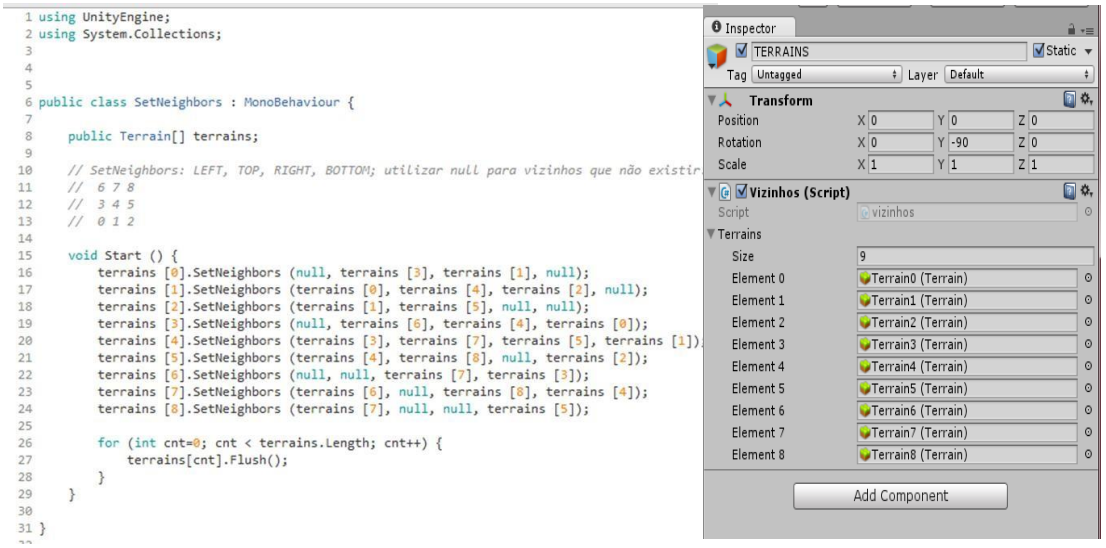


Figura A.28: Scritp em c# para junção dos terrenos.

Nesta versão o terreno está divido em 9 partes, sendo correspondido por uma matriz de 3x3, o procedimento será análogo ao anterior, só que neste caso não é necessário termos que estabelecer manualmente cada ligação com seu vizinho, basta somente criar um *game object* e colocar todos os terrenos que foram divididos dentro e estabelecer qual é a parte da matriz que pertence o referido terreno. Uma particularidade desses *scripts*, ele somente poderá ser aplicado onde os terrenos tem a mesma altura, ou seja, ele não trata as diferenças de alturas do terreno. Um exemplo da aplicação deste *script*, seria uma situação na qual tivéssemos vários terrenos planos e o objetivo final seria criar uma só superfície. A Figura A.29 mostra um exemplo onde temos 9 terrenos que foram interligados através do *script*.

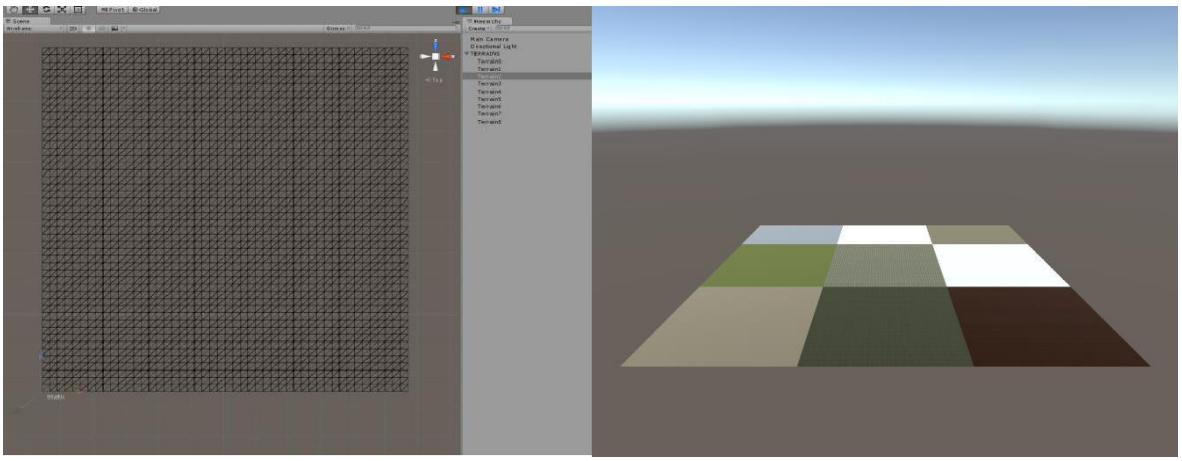


Figura A.29: Na parte direita da figura temos exemplo de 9 terrenos “colados” e na esquerda temos o *wireframe* respetivo onde aparecem todos terrenos interligados sem nenhum *gap*.

Anexo B: Abordagens da geração da animação dos personagens virtuais

Este anexo complementa a secção 4.3.2, personagens autónomos e possui como objetivo pormenorizar o processo de criação de animações dos personagens virtuais.

B.1: Criação da animação do modelo virtual feminino

Inicialmente não existe um controlador associado ao *Animator*. A criação de um novo *Animator Controller* é feita usando o menu *create* do separador *Project*, ou utilizando a opção como mesmo nome no menu *Assets*. É aconselhável que se coloque o ficheiro criado na pasta referente à personagem, para ser fácil de encontrar posteriormente. Por sua vez, a associação deste controlador à personagem é feita arrastando-a para o campo *Controller* do componente *Animator* (Figura B.1).

A definição do controlador é feita num editor específico, disponível como um separador Unity. É possível aceder a esse editor com um duplo clique sobre o controlador. Um controlador consiste numa máquina de estados. Quando o componente é iniciado, o controlador encontra-se no estado *Entry* e transita para o primeiro estado que lhe esteja ligado. O 1º passo corresponde à criação de um novo estado, clicando com o botão direito do rato sobre a área do editor e escolhendo a opção para adicionar um estado vazio *(Create State > Empty)*. Ao criar este novo estado a transição a partir do estado inicial é automaticamente criada e é apresentado o *Inspector* do estado acabado de criar.

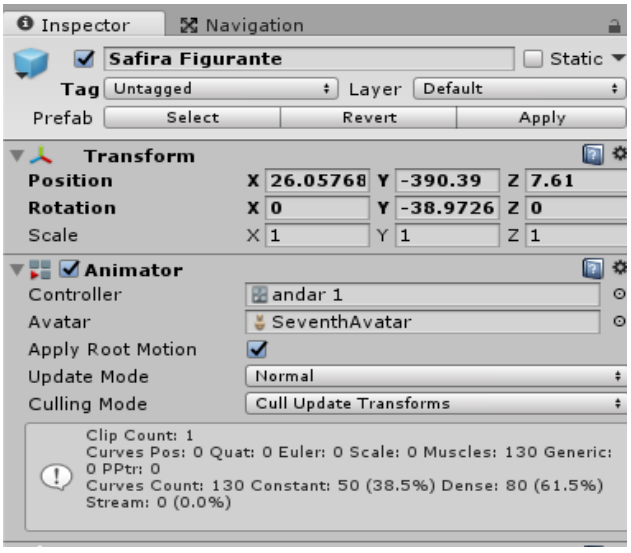


Figura B.1: Componente *Animator*.

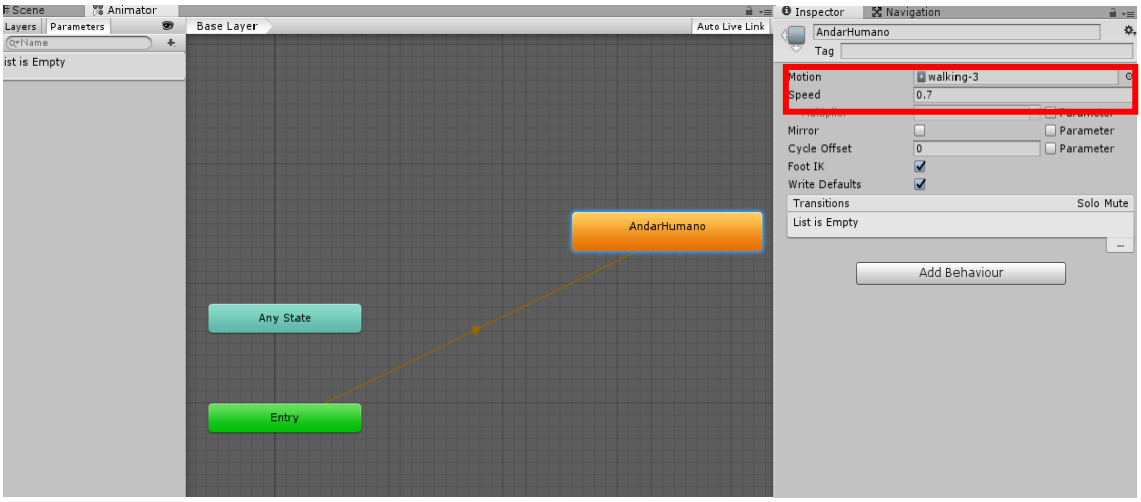


Figura B.2: Estado Andar Humano da personagem

Este primeiro estado irá representar a personagem em movimento. O nome do estado poderá ser alterado no *Inspector* respetivo. No campo *Motion* deverá ser associada a animação correspondente. A Figura B.2 apresenta o resultado destas operações. Infelizmente, a personagem rapidamente irá cansar de andar e vai parar, ficando estática. Se as animações quando são criadas não forem configuradas para funcionar em ciclo *(loop)*, o personagem virtual executa sua animação uma única vez e para. Para ficar em *Loop*, será necessário selecionar a animação no separador *Project* e, no *Inspector* respetivo, ativar as opções *Loop Time* e *Loop Pose* e usar o botão *Apply* para aplicar as alterações efetuadas (Figura B.3). Este processo foi aplicado para todas as animações utilizadas no projeto. A *opção Loop time* indica que a animação deverá ser repetida indefinidamente. Já a opção *Loop Pose* permite que o movimento seja mais fluido, não repetindo as posições iniciais e finais da animação.

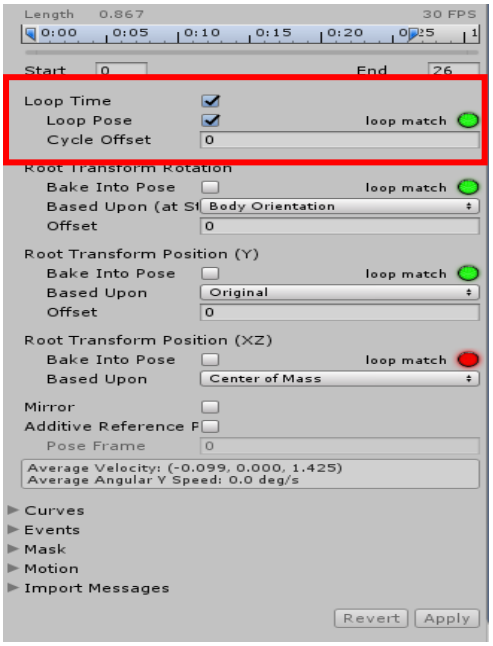


Figura B.3: Ativação das opções Loop Time e Loop Pose

De acordo com a Figura B.2, o estado A*ndarHumano* será executado depois do estado *Entry*, para que isso ocorresse foi preciso criar uma transição entre os dois estados, para tal, basta

clicar com o botão direito do rato sobre o estado de origem, e escolher a opção *Make Transition* e selecionar o estado de destino, para se criar um novo estado basta repetir o mesmo processo de criação do estado anterior sendo necessário associar uma animação nela. Da forma como foram criadas, as transições vão ser sempre executadas, já que não incluem quaisquer condições.

B.2: Criação da animação do burro

Os parâmetros podem ser de diferentes tipos. O menu de criação de parâmetros é apresentado na Figura B.4. Para o movimento, será utilizado o parâmetro *Trigger*. Os valores destes parâmetros podem ser alterados através de código e servir para controlar a transição entre diferentes estados.

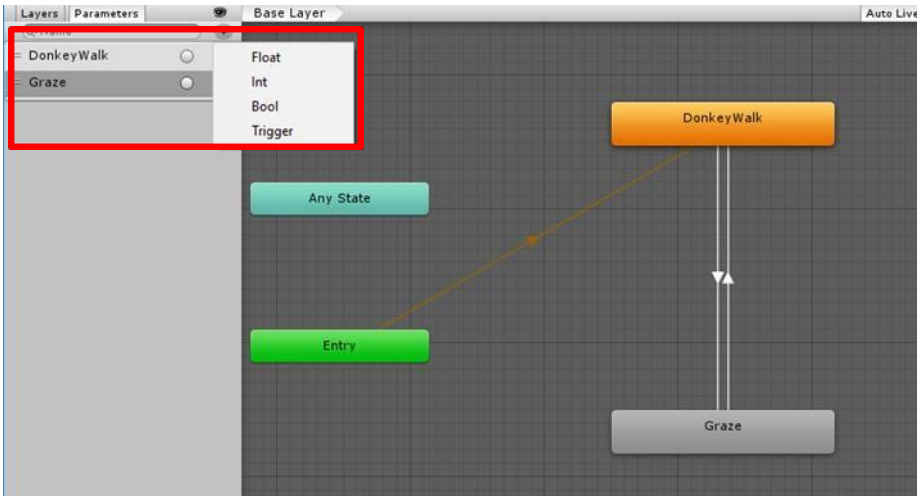


Figura B.4: Máquina de estados com os estados DonkeyWalk e Graze.

As transições são configuradas selecionando a respetiva transição e alterando as suas propriedades. Na Figura B.5 são apresentadas as propriedades das duas transições criadas. Cada uma destas transições necessita de uma condição para que a transição ocorra. As condições são adicionadas ao fundo do *Inspector*. No caso das transições em causa, as condições estão condicionadas a distância do objetivo.

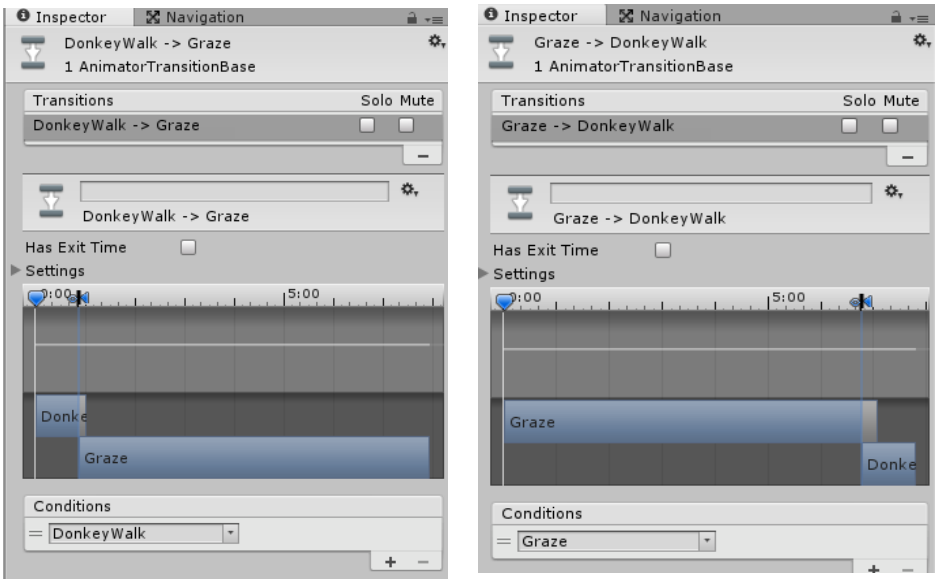


Figura B.5: Propriedades das transições entre DonkeyWalk e Graze,

Além das condições, é importante desativar a opção *Has Exit Time* em ambas as transições. Quando esta opção está ativa, a transição só será efetuada quando a animação do estado de origem tiver terminado. Com a opção desligada, a transição irá interromper a animação em curso, iniciando a animação do estado de destino. Existem certas animações que devem ser terminadas e não interrompidas como por exemplo: saltos.

Para que estas alterações tenham efeito, é necessário implementar uma *script*, denominada *chaseDonkey*, e associá-la ao burro.

```
2
3 using UnityEngine;
4 using System.Collections;
5
6 public class  chaseDonkey : MonoBehaviour {
7
8     public Transform player;
9     static Animator anim;
.0
.1     // Use this for initialization
.2     void Start ()
.3     {
.4         anim = GetComponent<Animator> ();
.5     }
.6
.7     // Update is called once per frame
.8     void Update ()
.9     {
:0
:1         if (Vector3.Distance (transform.position, player.position) > 2)
:2             anim.SetTrigger("Graze");
:3         else
:4         {
:5             anim.SetTrigger("DonkeyWalk");
:6
:7         }
:8     }
:9
:0 }
```

Figura B.6: Script responsável pelo movimento do burro.

Uma vez associado o *script* ao burro, vai permitir que ele caminhe até o objeto desejado ou um humano virtual, e quando a distância do objeto ou do humano virtual for menor que (parâmetro definido na *script*) a próxima animação será executada, até que a condição não se cumpra mais, ele voltará para o estado inicial, executando a primeira animação.

Tabela B. 1: Condições das transições entre os estados.

Origem → Destino	Condição
DonkeyWalk → *Graze*	Distância Obj < 2
Graze → *DonkeyWalk*	Distância Obj > 2

B.3: Controlo da Navegação sobre *NavMesh*

Na aba *Agents*, são definidos os diferentes tipos de agentes que serão controlados pela *NavMesh*, já na aba Areas permite definir diferentes tipos de áreas: aquelas por onde os personagens podem andar (*walkable*), não podem andar *(non walkable)*, e onde podem saltar *(jumpable)*. O uso de *NavMesh* pode ser em tempo real, em que a malha de navegação é calculada durante a execução do jogo, ou previamente, usando um processo habitualmente denominado cozinhar ou cozer (*bake)*.

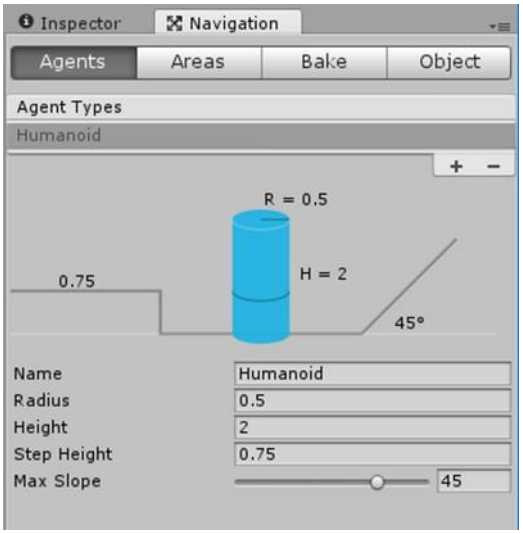


Figura B.7: Separador *Navigation*.

A Figura B.7 apresenta as dimensões dos personagens virtuais (humanos), pelo que o 1º passo que realizado foi definir as dimensões do agente genérico. É possível ajustar o tamanho do agente uma vez realizado o *bake*. O 2º passo realizado foi selecionar as zonas que eram permitidas caminhar. Uma vez terminado os passos anteriores, a *NavMesh* foi cozinhada.

B.4: Processo para dar comportamento autónomo aos personagens

O atributo da classe [CreateAssetMenu] é necessário para poder criar os *scriptable objects* através do menu contextual do Unity.

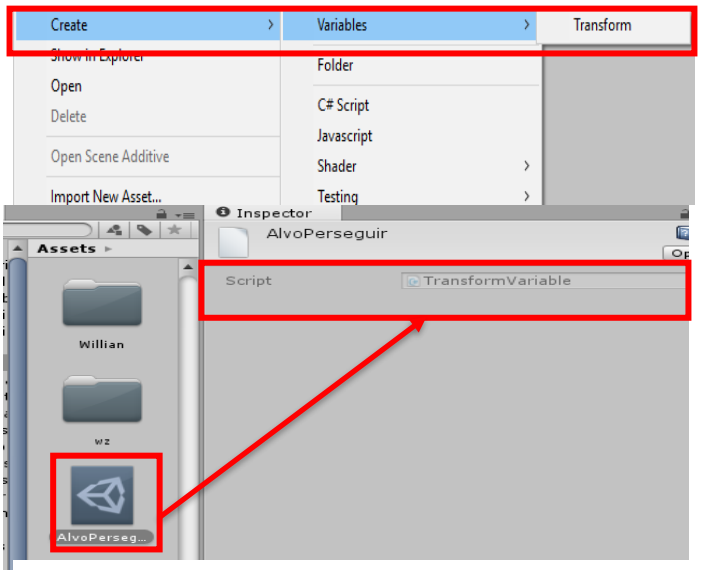


Figura B. 8: Processo de criação do scriptable

Ao pressionar o botão Transform, o *Scriptable Object* é criado, de acordo com a figura acima. Portanto o próximo passo será adicionar *Scriptable Object* criado (AlvoPerseguir) na referência do *target* (objetivo) do personagem que vai perseguir, que no nosso caso será o BurroAIPerseguidor.

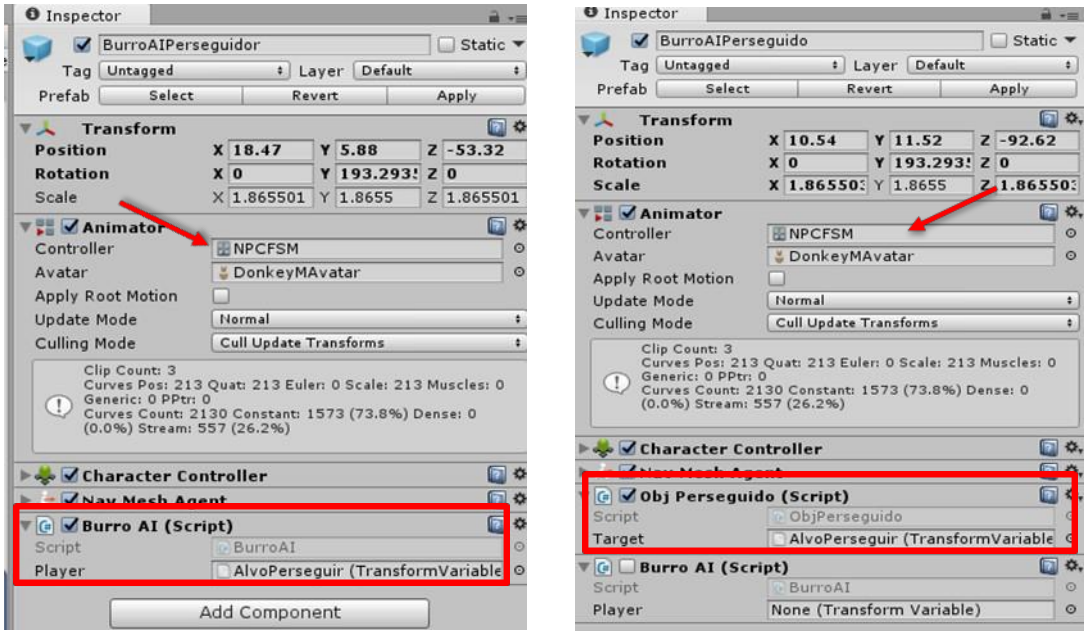


Figura B.9: A direita temos o Inspector do BurroAIPerseguidor e a esquerda do BurroAIPerseguido.

Como é possível observar na Figura B.9, para que BurroAIPerseguidor pudesse percorrer o objetivo assim buscar o alvo, foi criado um *script* chamado BurroAI. Já no *Inspector* do BurroAIPerseguido, foi criado um *script* chamado objPerseguido onde o *target* (objetivo) que deverá ser colocado será o *Scriptable Object* criado (AlvoPerseguir), com isso teremos a ligação entre o perseguidor e o perseguido.

É possível notar também que no objetivo (*Player*) do *script* BurroAI está vazio, sendo assim o BurroAIPerseguido, irá somente executar o percurso pré-determinado pelo (waypoints), e a certa altura entrará na rota do BurroAIPerseguidor, onde desencadeará uma perseguição e posteriormente um ataque. Para que isso funcione é extremamente importante adicionar o controlador da máquina de estados nos avatares correspondentes, sem ela não seria possível ter acesso as respetivas animações.

Uma vez completado este mecanismo o próximo passo, foi implementar o mecanismo do personagem Cavalo importado. Como explicado na secção 4.3.2, o cavalo adota duas animações Andar e Pastar, foi a criada a máquina de estados do cavalo análoga ao do Burro, porém era necessário criar um mecanismo que fizesse com que o Cavalo andasse com um objetivo, foram levadas duas alternativas, a primeira o cavalo teria um objetivo pré determinado, para isso foi criado um *script* chamado *finder*, que é responsável por encontrar o ponto de destino previamente selecionado ele utiliza a *NavMesh*, para o calcula da distância do objetivo, a outra alternativa foi criar um *script* na qual o cavalo percorria vários pontos dentro de uma zona, assim simularia outro tipo de comportamento, já que não ficaria estático ao chegar no objetivo.

```
using UnityEngine;
using System.Collections;
using UnityEngine.AI;

public class finder : MonoBehaviour {

    public Transform destinationPoint;

    void Update () {
        transform.GetComponent<NavMeshAgent> ().destination = destinationPoint.position;
    }
```

Figura B. 10: Pedaço de código do *script finder*.

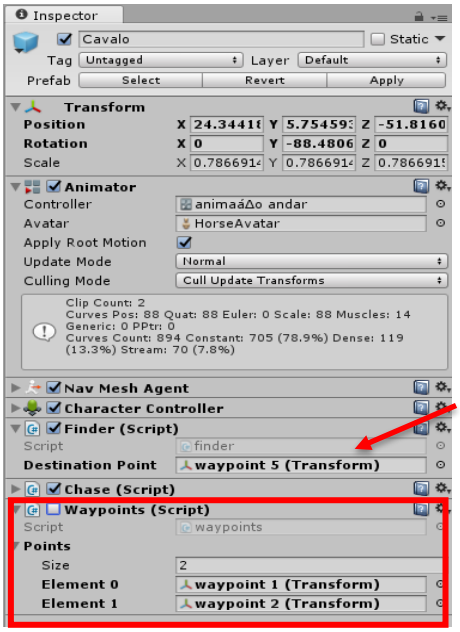


Figura B. 11: Inspector do cavalo com os *scripts* Finder, Chase e Waypoints criados

Uma vez finalizado o processo dos animais o próximo passo foi criar os *prefabs*[81]. *Prefab* é um tipo de *Asset* que pode ser entendido como um *GameObject* armazenado no Projecto. Um *prefab* pode ser inserido em diversas *Scenes*, diversas vezes. Quando é adicionado um *Prefab* a uma *Scene*, é criado uma instância dele. Todas as instâncias de um *prefab* estão conectadas com o *prefab* original. Quando um *prefab* é alterado todas suas instâncias, em todas as cenas, são alteradas automaticamente. São uteis para compartilhar elementos iguais entre diversas *Scenes.*

Para a criação dos *prefabs*, basta criar uma pasta nos *Assets* do projeto e arrastar o personagem desejado da *Hierachy* para essa pasta. Foi então criada a pasta chamada "Animais" e foram colocados todos os animais criados nela.

81 Prefabs: https://docs.unity3d.com/es/current/Manual/Prefabs.html